

Användarmanual för XAML-Tool

XAML-Tool är ett program som underlättar blankettproducing. Det genererar XML och XSD-kod från en XAML-fil och städar upp XAML-kod med några enkla klick. Går även använda för att snabbt generera ett träd, och annat diverse användbart vid blankettproduktion.

Innehållsförteckning

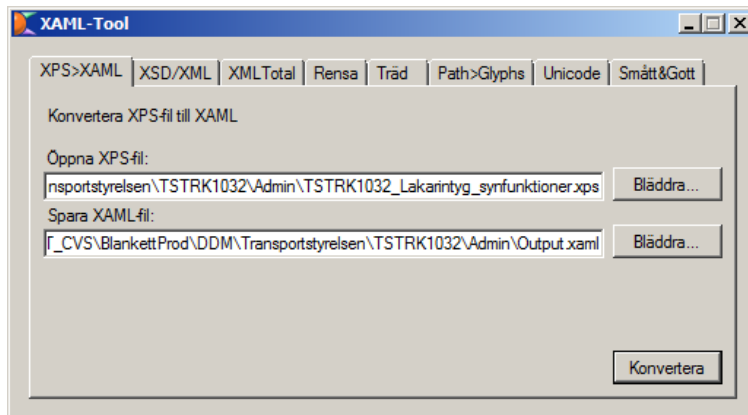
Beskrivning av programmet	2
XPS till XAML.....	2
Skapa XSD/XML.....	4
XMLTotal.....	5
Rensa XAML	7
Träd	9
Konvertera Path till Glyph.....	13
Unicode	17
Smått&Gott	18

Beskrivning av programmet

Programmet består av åtta delar, indelat i flikar. Första fliken gör om en XPS till Forms-utformad XAML. Andra fliken skapar XSD- och XML-kod för formlist till din blankett. Tredje skapar ett helt komplett XML. Den fjärde fliken rensar en XAML-fil från onödiga attribut och liknande. Den femte fliken används för att skapa trädet till din blankett. I den sjätte fliken kan du göra om path-bokstäver till Glyphs! På den sjunde fliken kan du konvertera felaktiga UnicodeStrings. På den åttonde fliken finns lite allt möjligt användbart.

XPS till XAML

I XPS-dokument finns kod som liknar XAML väldigt mycket, och som för det mesta går att använda rakt av. Däremot är typsnittsreferenserna helt fel, och ibland används externa resurser som är jobbigt att ersätta manuellt. Med det här verktyget sköts allt sånt automatiskt, och resultatet blir en XAML som är utformad för Forms, där du bara behöver justera några småsaker och lägga in fält.



Öppna XPS-fil

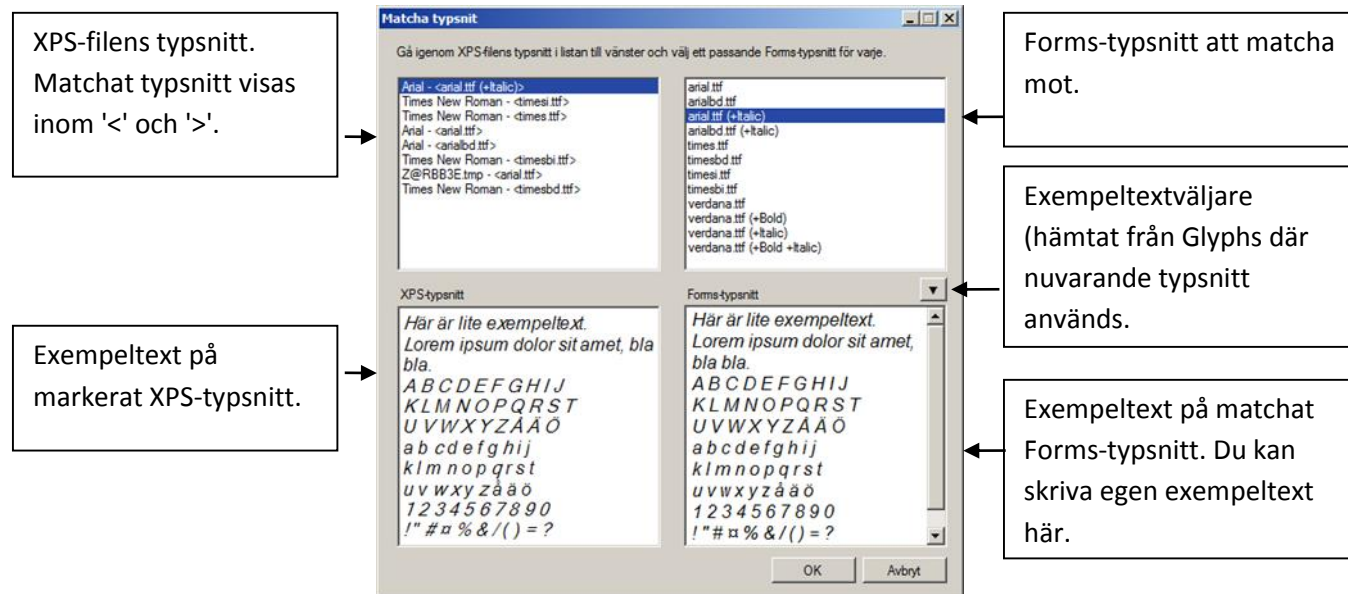
Här väljer du sökväg till blankettens originallayout i XPS-format. Om du bara har en Word-fil eller PDF, så kan man göra om dessa till XPS genom att skriva ut dem med skrivaren "Microsoft XPS Document Writer".

Spara XAML-fil

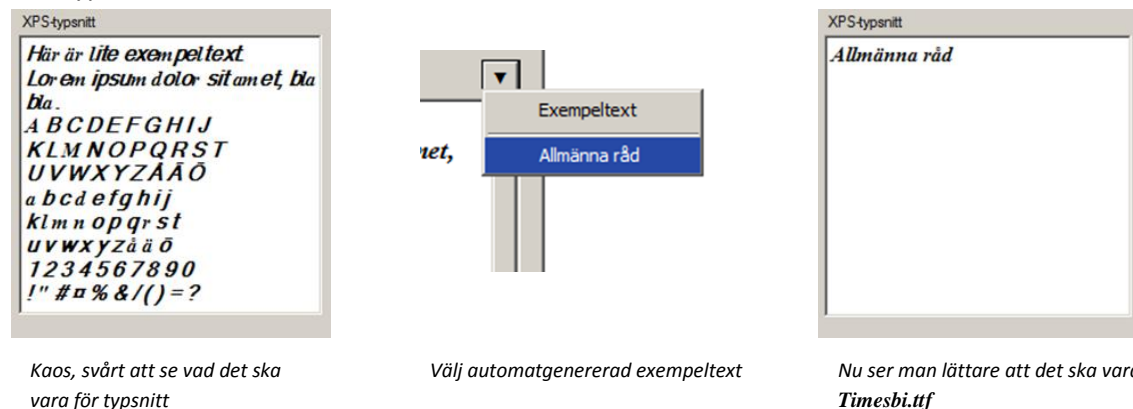
Välj sparplats för resulterande XAML-fil.

Matcha typsnitt

Programmet kan behöva din hjälp med att matcha typsnittsreferenserna mot ett passande Forms-typsnitt. XAML-Tool kan ofta gissa sig till rätt typsnitt, men i vissa fall går det inte, och då kommer programmet sätta dessa till Arial. Du kan dock ändra och granska typsnittskopplingen via ett gränssnitt.



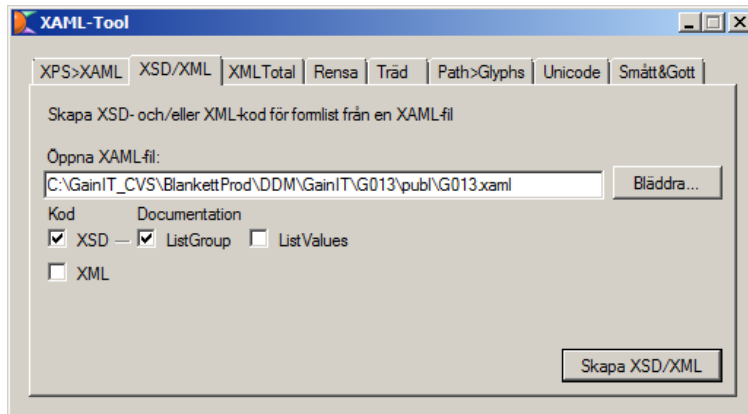
XPS-typsnittet kan ibland se lite märkligt ut, blandat med Arial och sig självt. Detta beror på att XPS-typsnittet inte innehåller alla tecken, utan oftast bara de som används. Det kan då vara fördelaktigt att använda exempeltextväljaren för att få fram exempeltext som endast innehåller tecken som används av XPS-typsnittet.



OBS: Om XPS-filen innehåller bitmapsbilder kommer dessa att ersättas med en färgglad ruta där bilden skulle varit. Detta får du sedan ersätta med en vektorbaserad version av bilden.

Skapa XSD/XML

På denna flik skapar du XSD-kod och/eller XML-kod för formlist som du sedan ska klistra in i respektive fil för blanketten. Koden för XML är lite halvt obsolet, då det finns XMLTotal. Mer om det längre ner.



Öppna XAML-fil

Här ska du välja blankettens XAML-fil som du vill generera XML och XSD-kod från. Klicka på "Bläddra..." för att lättare specificera sökväg. Textrutan kommer automatiskt att fyllas i med sökväg till den fil du valde.

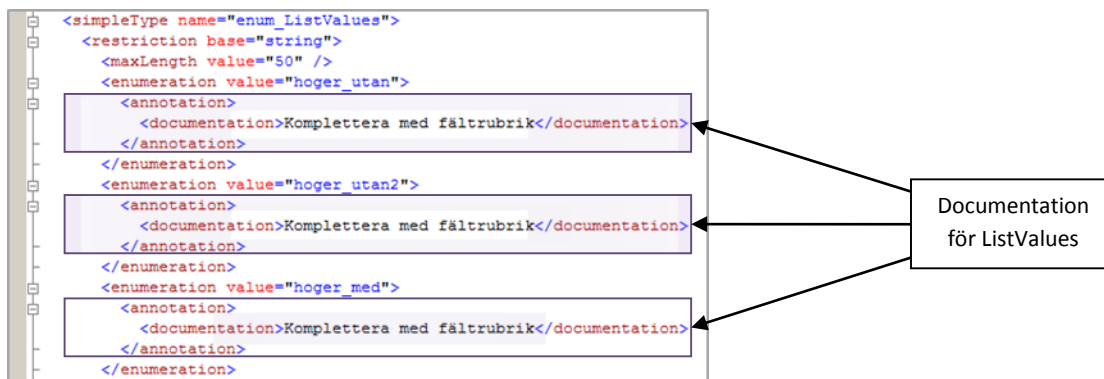
Kod

Här väljer du vad för kod som ska genereras. XSD är förvalt eftersom XML-koden ingår i XMLTotal, men om du gärna vill ha XML-kod endast för formlist så är det bara kryssa i denna.

Documentation

Om någon av dessa är ikryssad kommer dokumentationstaggar att skapas för respektive kod (ListGroup/ListValues) i XSD-delen. XML har inget sånt.

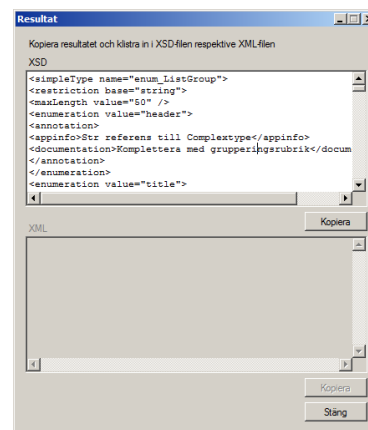
Ofta är detta bara nödvändigt för ListGroup. ListValues är ofta för många för att bry sig om.



När du klickar på "Skapa XSD/XML" så får du upp ett nytt fönster som innehåller de genererade koderna.

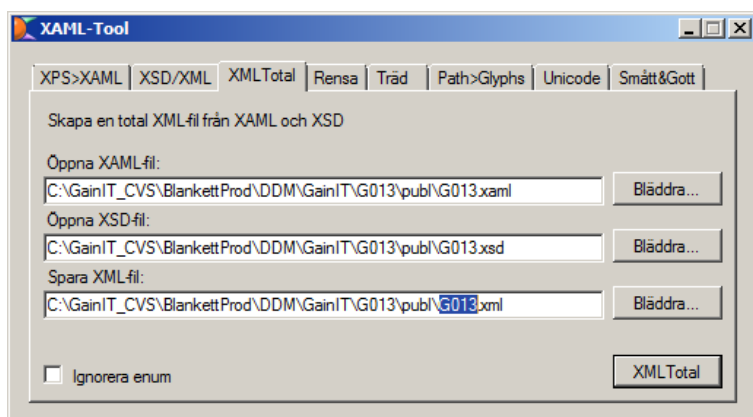
Innehållet i XSD klistras in i blankettens XSD-fil. Du ska ersätta de befintliga taggarna `<simpleType name="enum_ListGroup">` och `<simpleType name="enum_ListValue">` och deras innehåll.

Innehållet i XML ska du klistra in i din blanketts XML-fil. Ersätt hela `<formlist>`-taggen.



XMLTotal

Här kan du skapa ett komplett XML till din blankett! Nu behöver du inte längre trilskas med xpaths och annat. Låt XAML-Tool göra det istället! Allt du behöver är din XSD-fil och XAML-fil.



Öppna XAML-fil

Som vanligt ska du skriva in sökvägen till en XAML-fil. Använd gärna knappen "Bläddra...". Gör du det kommer de andra fälten fyllas in automatiskt, baserat på det filnamn du angav.

Öppna XSD-fil

Samma som ovan, fast XSD (schemat).

Spara XML-fil

Välj var du vill spara filen. Resultatet är alltså en XML-fil, redo att användas till din blankett. Programmet kommer att skriva över filer som redan finns, så var försiktig!

Ignorera enum

Om denna är ikryssad så kommer programmet strunta i att vissa xpaths i XAML-filen inte har giltig enum. Om schemat säger att det finns "Primär" och "Sekundär" att välja på, men XAML-filen ändå antyder att det finns något som heter "Annat" så kommer "Annat" att hamna i XML:et även fast det inte är helt giltigt enligt schemat. Detta fungerar, men är inte rekommenderat.

OBS: Endast attributvärdet kan ignoreras, själva attributet måste finnas! [`@typ='bla'`]

När du specificerat sökvägar till alla filer klickar du på "XMLTotal". Om allt står rätt till kommer en XML-fil skapas på den plats du angav.



Du kommer även få upp en lista på xpaths som inte stämmer mot schemat, om det finns några. Det kan bero på följande:

- Xpathens syntax är felaktig. Det kan saknas en apostrof eller en hakparentes någonstans.
- Platsen finns inte i schemat. Kolla vilka enums som finns i xpathen och kolla om motsvarande enum finns i schemat. För formlist, se till att köra XSD/XML (ovan) innan du genererar ett XML.
- Det är ingen xpath. Programmet försöker hämta alla xpaths från XAML-filen. Det kan hända att det hämtar även sånt som inte är xpaths. PrintMode är undantaget, men det kanske finns något annat? I så fall kan du ignorera varningen.

OBS: Numeriska identifierare (såsom "adressED1[1]/blabla") har inte fullt stöd. Om blanketten använder sådana måste du kontrollera resultatet noga. Det ska fungera bra om de ligger i rätt ordning i XAML-filen ([1], [2], [3]) men inte om [3] kommer före [1], t.ex.. Programmet ger heller ingen varning kring dessa. Det är rekommenderat att du använder attributidentifierare ("[@typ='rad1']") om det går.

Programmet verkar inte hitta några xpaths i XAML-filen? Det kan bero på att Tag-attributet inte är i sin egen tagg. Flytta i så fall ut Tag så att alla blir glada.

Tips! Använd [Rensa XAML -> Flytta ut Tag](#).

<code><TextBox x:name="exempel" blablabla="blabla" Tag=" patientED1/blabla converter " /></code>	
<code><TextBox x:name="exempel" blablabla="blabla" > <TextBox.Tag> patientED1/blabla converter </TextBox.Tag> </TextBox></code>	

An arrow points from the Tag attribute in the first row to the Tag element in the second row.

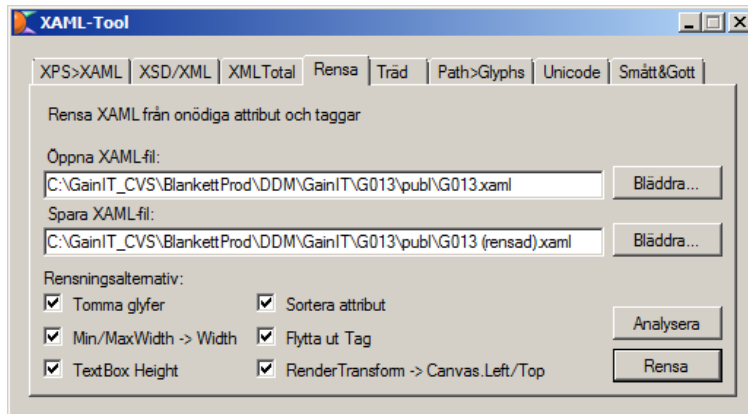
Eventuella problem

Att tänka på vid autogenerering av XML är att funktionen enbart tar hänsyn till det som enligt schemat ska finnas, och det som används av XAML. Därför bör du inte lita totalt på funktionen vid generering av XML till ett eBrev, t.ex., som även ska ha andra taggar i XML:et för att fungera (attachments, consignment_1, bgpgocrs o.s.v.). Eftersom dessa taggar inte förekommer i XAML eller är krav enligt schemat kommer dessa uteslutas.

Ett tips är att använda funktionen och spara till en annan fil, t.ex. "blankettnummer temp.xml" och sedan kopiera över innehåll från den och se till att inte ersätta eBrevsspecifika taggar.

Rensa XAML

På denna flik finns olika alternativ och möjligheter för att skapa en finare XAML-fil, rensad från skräp och onödiga attribut.



Öppna XAML-fil

Här väljer du en XAML-fil som du vill rensa. Klicka på "Bläddra..." för att lättare få rätt sökväg till den fil du vill använda.

Spara XAML-fil

Här skriver du in sökväg och namn till filen som ska sparas. Det kan vara samma som filen som öppnas men det är rekommenderat att du använder ett annat namn så att originalet inte sparas över.

Alternativt kan göra en säkerhetskopia av original-XAML-fil, ifall något går snett...?

Sen får du välja vad i filen du vill städa upp. Här följer en förklaring på varje alternativ:

Tomma glyfer

Tar bort <Glyphs ...> där UnicodeString inte innehåller några synliga tecken, som bara mellanslag eller ingenting. Rättar även till strängar som påbörjas och/eller avslutas med mellanslag, t.ex. " hej " blir till "hej". I de fall då mellanslag tas bort i början av strängen kommer glyphen flytta sig åt höger för att kompensera för mellanslagens bredd.

Innan: " Här är exempeltext "

Efter: "Här är exempeltext"

OBS: Om en Glyphs är inom en speciell StackPanel för dynamisk position på blanketten (som flyttar sig när en textlåda över får fler rader), så kan en osynlig Glyphs ibland användas som tom rad. Dessa kommer tyvärr att rensas bort.

Min/MaxWidth -> Width

Om MaxWidth och MinWidth är samma så görs de om till Width. Detta gäller för TextBox, CheckBox och TextBlock.

Om elementet innefattas av en WrapPanel så händer ingenting, eftersom man ibland vill ha fast bredd på ett element (t.ex. datum) vilket uppnås genom att ange samma Min/MaxWidth.

TextBox Height

"Height" tas bort från TextBoxar. Varför? För att TextBoxars höjd ska avgöras av antal rader för att undvika att det blir halva rader. Var uppmärksam på om detta rensas bort då du kan behöva lägga in MinLines och MaxLines istället där det behövs i din XAML-fil.

Sortera attribut

Detta sorterar attributen i TextBox, CheckBox och TextBlock efter en speciell ordning. Detta kan vara bra om du vill ha ordning på var du har Width och liknande, så det blir lättare att hitta om man letar.

För tillfället är det följande ordning:

"x:Name", "Name", "IsEnabled", "ToolTip", "MaxLength", "Style", "TextWrapping", "AcceptsReturn", "Validation.ErrorTemplate", "RenderTransform", "MinLines", "MaxLines", "MinWidth", "Width", "MaxWidth", "Margin", "Canvas.Left", "Canvas.Top".

Attribut som inte finns med i listan hamnar i slutet.

Flytta ut Tag

På vissa gamla blanketter ligger Tag-attributet inte i sin egen undernod. Detta kan vara mycket jobbigt då det blir långa rader och svårt att hitta bland attributen och sådär. Dessutom fungerar inte XMLTotal på detta sätt.

Vad denna funktion gör är att den flyttar ut Tag från element och sätter det som en egen undernod till elementet.

RenderTransform -> Canvas.Left/Top

Ofta när en PDF skrivits ut till XPS så kommer RenderTransform att ha använts för att positionera Glyphs och annat, och ibland är det en kombination av RenderTransform och OriginX/Y och FontRenderingEmSize, vilket kan vara ganska opraktiskt att jobba med, och även onödigt.

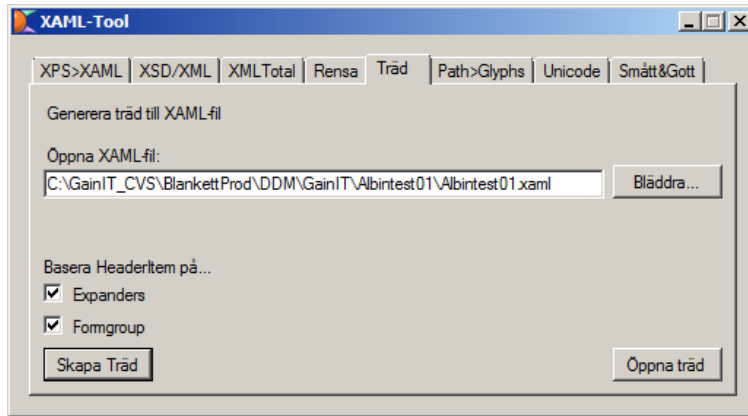
Denna funktion kommer att göra om alla RenderTransform till OriginX/Y och FontRenderingEmSize för Glyphs, och Canvas.Top/Left för checkboxar, textboxar och textblock. Detta kommer dock bara fungera om RenderTransform har samma skalningsproportioner för X och Y och ingen snedvridning (eller vad man ska kalla det), ungefär såhär: `RenderTransform="S, 0, 0, S, X, Y"` (S = skala, X och Y = position). T.ex. så kommer viss kursiv text inte att påverkas av detta (använder "snedvridning").

När du har bestämt dig för exakt vad du vill rensa/städa så klickar du på "Rensa". En fil kommer sparas med det namn du angav. Du får även en liten rapport på hur mycket som rensats.

Om du endast vill ha en rapport utan att spara någon fil klickar du på "Analysera". Detta är bra om du bara vill kolla om filen är OK eller om den måste rensas, eller om det är väldigt lite som behöver rensas så kan du kanske göra det själv.

Träd

Detta verktyg är byggt för att underlätta trädskapandet av en blankett. Du kan generera ett nytt träd eller öppna ett befintligt träd för redigering.



Skapa träd

Med det här alternativet kommer ett helt nytt träd genereras. Det finns två val du kan göra för att påverka hur trädet genereras.

Expanders: Varje expander i XAML-filen blir till ett HeaderItem. Programmet förväntar sig att expandern har ett namn (Name eller x:Name) och ett TextBlock inom Expander.Header.

Formgroup: Genereringen går efter x:Name och xpath (<*.Tag>). Resultatet beror helt på hur bra namnen är i blanketten.

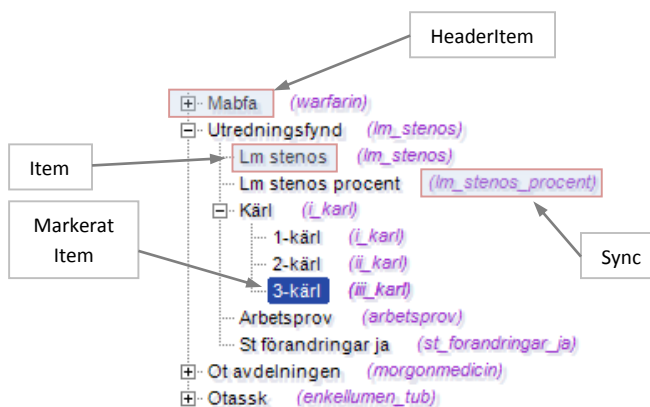
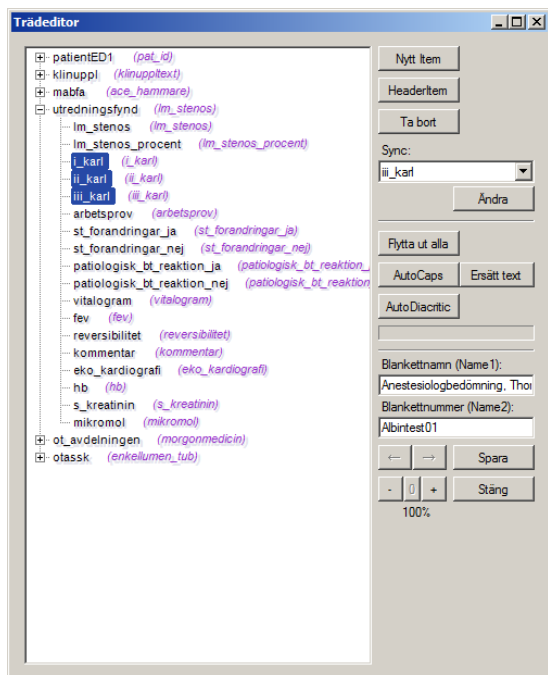
Det går att kombinera dessa utan problem.

Öppna träd

Välj en XAML-fil eller en txt-fil och klicka på "Öppna träd". Detta kommer öppna ett träd som finns i en XAML-fil eller textfil. Det som krävs är att filen är giltig XML och innehåller en <RootHeaderItem>.

Redigering

När du har valt ditt sätt att öppna trädeditorn kommer du se en ruta som innehåller ett gäng knappar och en trädstruktur. Nu ska du göra ändringar i detta träd!



Här ser du ett exempel på hur ett genererat träd kan se ut. Ett plustecken bredvid en nod betyder att det är ett HeaderItem och innehåller fler noder, som i sin tur kan vara HeaderItem eller Item. Den lila/rosa texten visar vad noden har för Sync. Sync är dit Forms hoppar när man klickar på texten i trädet, men det visste du nog redan.

Noder längst ut, även de som inte har några undernoder, blir automatiskt till HeaderItem.

Markera noder

Det finns många sätt att markera noder.

Klick: Markerar noden du klickade på. Alla andra avmarkeras.

Ctrl+klick: Markerar flera noder eller avmarkerar en nod om den redan är markerad.

Shift+klick: Markerar alla från och med den först markerade noden till och med den du klickade.

Shift+pil upp/ner: Markerar uppåt/neråt.

Pil upp: Flyttar markering till noden ovanför den senast markerade noden.

Pil ner: Flyttar markering till noden neranför den senast markerade noden.

Pil vänster: Flyttar markering till övernoden till den senast markerade noden. Stänger även öppna noder.

Pil höger: Öppnar noder eller flyttar undernoden.

Ändra nod

Namn: För att ändra namn på en nod dubbelklickar du på den. Det går även bra att ha den markerad och trycka Enter eller F2. Skriv in det namn du vill ha och tryck Enter eller klicka utanför. Tryck Esc om du helt plötsligt inte vill ändra nodens namn längre.

Sync: Markera en eller flera noder och välj i listan till höger den Sync du vill ha, och klicka sedan på "Ändra" precis under, eller tryck Enter. Du bör se att texten bredvid namnet på noden/noderna ändras.

Tips: Använd TAB för att växla fokus till listan till höger, och tryck Enter för att spara Sync på noden. Fokus kommer att hamna på trädet igen.

Flytta nod

Det finns två sätt att flytta på noder.

Dra-och-släpp: Det vill säga att du håller muspekaren över en nod, trycker in en musknapp, flyttar pekaren dit du vill ha noden och släpper. Du kommer att se en markör där noden kommer hamna. Om du vill flytta in noden i en annan så släpper du noden i rutan som kommer fram till vänster (☐).

Ctrl+Piltangenter: Ha en eller flera noder markerade. Håll inne ctrl och tryck upp för att flytta noden ett steg upp, ner för ner, vänster för att flytta ut noden från sin övernod och höger för att flytta in noden i noden nedanför.

Lägg till/Ta bort nod

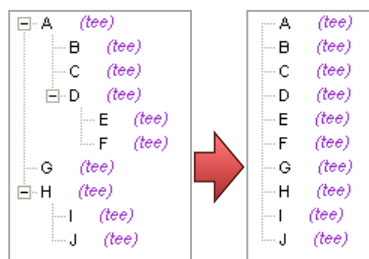
Lägg till: Klicka på knappen "Nytt Item" så kommer en nod skapas där du har markeringen. Ändra namn och Sync, så är det klart!

Gruppera: Markera flera noder och tryck "HeaderItem" så kommer alla markerade noder hamna i ett nytt HeaderItem som du får namnge. Glöm inte ändra Sync!

Ta bort: Markera en eller flera noder och tryck på "Ta bort" eller delete. Alla markerade noder och dess undernoder kommer att tas bort.

Flytta ut alla

Om du inte tycker om trädets struktur men vill ha kvar noderna som finns där för att skapa din egen struktur från grunden kan du trycka på denna knapp. Vad som kommer hända är att alla noder kommer att hamna löst i trädet, alltså inga noder kommer innehålla andra noder.



AutoCaps

Ofta är noders namn helt OK förutom att de inte börjar med stor bokstav. Ett smidigt sätt att lösa detta är att använda AutoCaps-funktionen. Klicka på knappen där det står "AutoCaps" så får du upp en meny där du får välja hur trädets noder ska versaliseras.

Första menyvalet gör så att alla noders namn kommer att få sin initial versaliserad.

"utredningsfynd" → "Utredningsfynd"

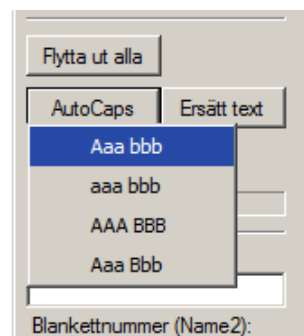
"insulin" → "Insulin"

"arbetsprov" → "Arbetsprov"

Andra gör om alla bokstäver till gemener, tredje gör om dem till versaler och det fjärde gör så att varje bokstav som direkt följer ett mellanslag blir en versal. Alla bokstäver som redan är versaler behåller sin versalstatus.

När du gjort ett val får du upp en bekräftelseruta där du måste klicka på "Ja".

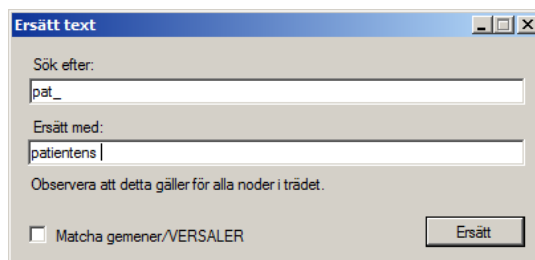
OBS: Om du vill göra om t.ex. "MOTORIK" till "Motorik" måste du först göra om alla bokstäver till gemener, sedan välja första menyvalet (Aaa bbb) eller fjärde (Aaa Bbb).



Ersätt text

Med denna funktion kan du ersätta text som förekommer i trädnode med en annan text. Om du vill göra om understreck till mellanrum eller "pat" till "patient" så kan du göra det här.

Kryssa i "Matcha gemener/VERSALER" för att göra sökningen skiftlägeskänslig.



AutoDiacritic

Trött på alla "vardenhet", "lasformaga" och "forfragan"? Klicka på AutoDiacritic så kommer XAML-Tool att fixa till detta för alla noder i ditt träd! Fungerar oftast jättebra, men kontrollera gärna! Det är t.ex omöjligt för XAML-Tool att veta om "ar" ska vara "år" eller "är", så ibland kan resultatet bli lite galet... Men det är bara rätta till!

Blankettnamn och Blankettnummer

Innan du sparar bör du kontrollera dessa.

"Blankettnamn" är baserat på första största Glyphs i XAML-filen. Oftast är detta rubriken, det som ska vara blankettnamnet, men ibland kan det vara en streckkod eller en dum text där det står "Välkommen!" i stora bokstäver. Notera också att rubriken väldigt ofta är på två rader, och om så är fallet kommer texten bara innehålla första raden.

"Blankettnummer" är rätt och slätt baserat på filnamnet minus filändelse (.xml), så detta behöver sällan ändras.

Ångra/Upprepa

Om du gjort ett misstag klickar du på knappen med en pil som pekar åt vänster (←) så kommer trädet återgå till ett tidigare stadie. Om du vill ha tillbaka misstaget trycker du på knappen med en högerpil (→).

Det går även bra att använda kortkommando Ctrl+Z för att ångra, och Ctrl+Y/Ctrl+Shift+Z för att upprepa.

Zoom (förstora/förminska)

Knapparna + och - (samt 0) används för att förstora/förminska texten i trädet. Det går även bra att använda Ctrl-Scroll, och Ctrl + +/- . Knappen 0 används för att återställa storleken till ursprungsläget. Det går även att använda Ctrl + 0.

Spara

När du är nöjd med ändringarna klickar du på "Spara". Då får du upp en ruta med det genererade trädets XML som du kan klistra in i din blankett. Du kan även spara till fil genom att klicka på "Spara som...." så får du upp en ruta där du kan ange sparplats och namn. Klicka på OK och filen sparas. Nu kan du välja om du vill öppna filen i det program du har associerat till den filtypen (troligtvis Notepad eller Notepad++). Kopiera nu innehållet i filen och klistra in i din XAML-fil där trädet ska vara.

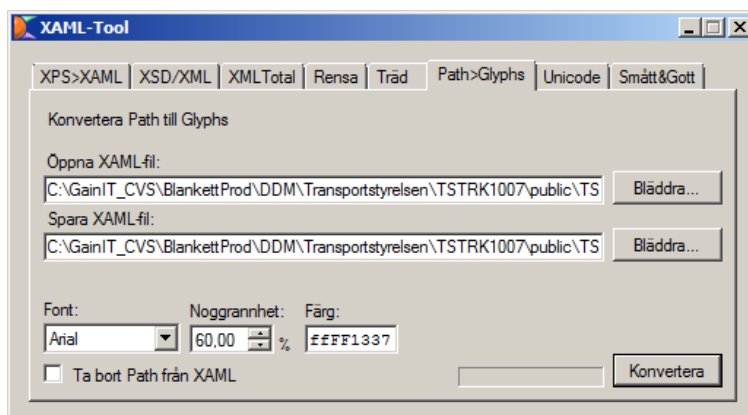
Konvertera Path till Glyph

Ibland är blankettens text inte bokstäver utan bara linjer och kurvor. Detta är väldigt opraktiskt då det tar upp väldigt mycket onödig plats vilket medför att blanketten tar längre tid att ladda. Dessutom är det inte smidigt att hantera då det inte går markera någon text och så vidare.

Att byta ut alla Pathar i en blankett till Glyphs manuellt kan ta väldigt lång tid.

Istället för att manuellt sätta in nya Glypher istället för Path kan du låta XAML-Tool göra detta åt dig.

Resultatet varierar och blir nästan aldrig helt perfekt, men är ofta en bra start.



Öppna XAML-fil

Skriv in sökväg till den XAML-fil som du vill hantera.

Spara fil

Välj var filen ska sparas.

Font

Välj från listan det typsnitt som ska kollas efter. Olika typsnitt ser olika ut, och de skiljer sig från blankett till blankett.

Arial ser ut såhär
Times ser ut såhär
Verdana ser ut såhär
Ocrb10 ser ut (ungefär) såhär

Många tecken liknar varandra mellan fonter, som punkt, bindestreck och kolon. Om blanketten har flera olika fonter kan dessa tecken tolkas fel.

Nogrannhet

Här specificerar du hur exakta resultaten ska vara. Lägre nogrannhet innebär större chans att en Path identifieras som en bokstav. Vid 0% kommer alla Pathar bli till Glypher och blanketten kommer se helt mupp ut. 60% är min rekommendation, men om du tycker att den får med för lite/mycket måste du justera detta värde.

Färg

Här skriver du in den färg som resulterande Glyphs kommer få (i attributet "Fill"). Färgen formateras enligt standard hexadecimal formatering och programmet säger till om du gjort fel. Det kan vara bra att ha en annan färg än de Pathar som ska konverteras, så att man kan se om resultatet är OK och var man kan behöva rätta till. När du är klar med justeringen och alla onödiga Pathar är borttagna är det lätt att sedan ersätta färgen med svart färg.

Sökandens personnummer

*Här kan jag tydligt se att fontstorleken
behöver justeras lite*

Ta bort Path från XAML

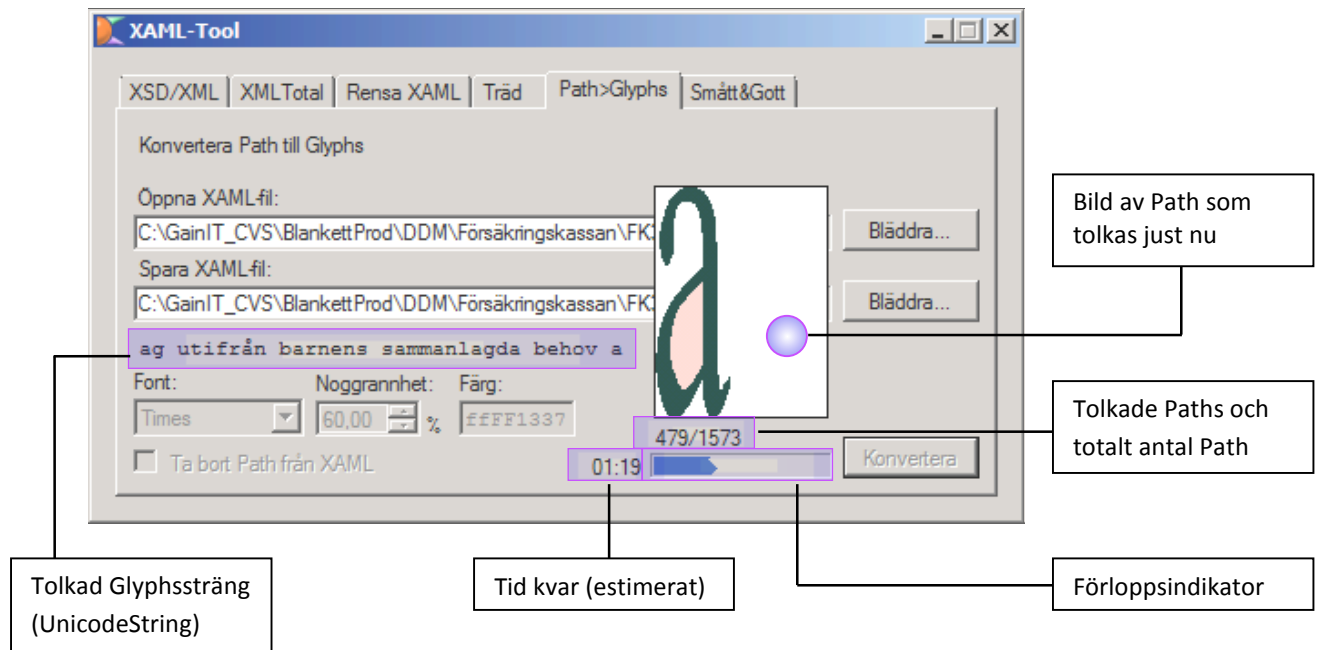
Om denna är ikryssad kommer de paths som identifieras som bokstäver att tas bort från den resulterande XAML-filen. Övriga är orörda.

Ibland vill man kanske kontrollera resultatet. Du får i så fall ta bort patharna manuellt.

All Glyphkod kommer att hamna innan första Pathkoden, så ett sätt att jämföra är att ersätta alla glyphfärger med något unikt och synligt som "#fff1337". Nu kommer röd text synas bakom original-Path, så nu får du jämföra med ögat.

När programmet kör

När du har tryckt på "Konvertera" och allt står rätt till så sätter XAML-Tool igång med att försöka tolka alla Pathar. Under tiden ser du status i form av estimerad tid kvar och tolkade pathar/totalt antal.



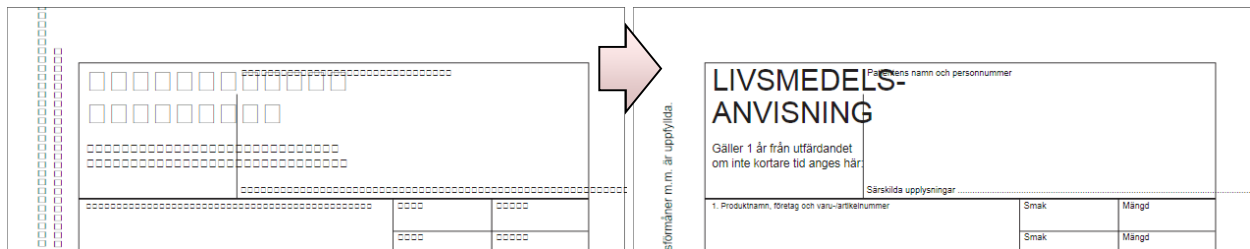
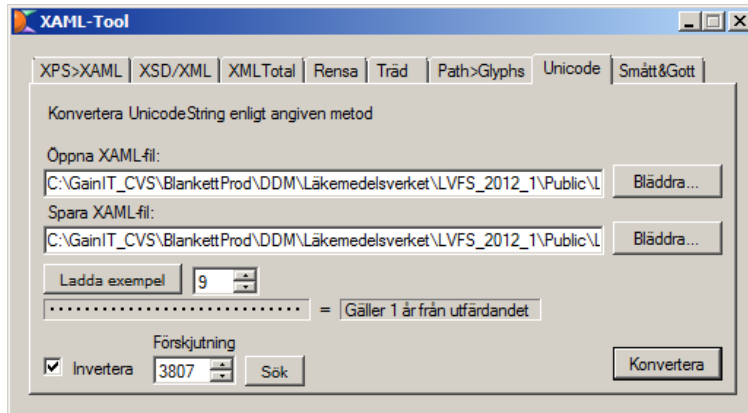
Problem som kan uppstå

Som sagt är denna konvertering inte helt perfekt. Här är några av de problem som kan uppstå:

- 'I' (stora i) och 'l' (lilla L) ser nästan exakt likadana ut för Arial och kommer i de flesta fall tolkas som 'I' (stora i). Ibland kan även 'i' (lilla I) tolkas som stort i eller litet L.
- För bokstäver vars gemenform liknar versalformen, fast mindre (som Cc, Oo, Pp, Ss Vv Ww, Xx och Zz) är det stor chans att det blir fel.
- Punktlister kan vara problematiska då själva punkten i punktlistan kommer tolkas som en punkt (som finns i slutet av en mening).
- Vissa tecken har uteslutits, som t.ex. », ¶, ¥ och ¿. Detta för att effektivisera och för att vissa kan blandas ihop med andra bokstäver. I de fall då de faktiskt förekommer på blanketten kommer programmet missa dessa.
- Fontstorleksuppskattningen är inte perfekt och kan bli för stor eller för liten, och är sällan konsekvent. Blanketten kommer ha ungefär lika många fontstorlekar som Glyphs. Om fler än två tecken misstolkas kan fontstorleken påverkas.
- Textdekorationer (som **fet** och *kursiv*) kommer inte kännas igen.
- Loggor och linjer/rutor är också Path, och kommer att försöka tolkas av programmet och skapa skräp.
- Det kan ta lång tid för programmet att utföra konverteringen. Under tiden kan du göra något annat, som t.ex. hämta kaffe, rita, vika pappersflygplan och/eller jobba på andra delar av blanketten.
- Roterad text går inte tolka. Kan ibland tolkas felaktigt, t.ex. ett roterat I som ett understreck.
- Ibland kan avståndet mellan ord missbedömas och felaktiga mellanrum kan skapas.
- Om en bokstav är uppbyggd av flera Pathar kommer det bli fel, t.ex. om ett 'O' är en stor svart boll med en mindre vit boll framför. 
- Om patharna kommer i omvänd ordning i XAML-filen kommer varje bokstav bli en egen Glyphs då programmet bara kollar om nästa Path ligger till höger för att se om de hör ihop.
- Ibland, av okänd anledning, kan programmet gå långsammare än vanligt. Vanlig hastighet är ungefär 12 paths per sekund, men ibland går det 4 paths per sekund. Händer detta går det oftast snabbare om du startar om programmet och försöker igen.

Unicode

Ibland när man skrivit ut sin XPS så är alla UnicodeString bara massa rutor (t.ex. ?????????). Det går att fixa detta manuellt genom att helt enkelt kolla på blanketten vad det ska stå och sedan skriva, eller sök/ersätt bokstav för bokstav. XAML-Tool kan göra detta åt dig!



Öppna XAML-fil

Här fyller du i sökvägen på den XAML-fil som ska hanteras.

Spara XAML-fil

Önskad sökväg till den konverterade filen.

Ladda exempel

När du klickar här så kommer alla UnicodeStrings hämtas från den angivna XAML-filen, så kan du se hur resultatet kommer att bli. Det måste vara en giltig fil angiven i Öppna XAML-fil.

Använd bläddraren till höger om knappen för att byta UnicodeString.

Under kan du se hur UnicodeString ser ut nu, och hur den kommer se ut när den är konverterad.

Invertera

Oftast är byte-ordningen inverterad, men inte alltid. Om du inte hittar en förskjutning som passar till XAML-filen så kan du prova att kryssa ur denna.

Förskjutning

Här anger du förskjutning på konverteringen. Om Invertera är ikryssad blir det inte riktigt lika konsekvent, men med den urkryssad kan du tydligt se att 'a' blir till 'b' om förskjutningen är 1.

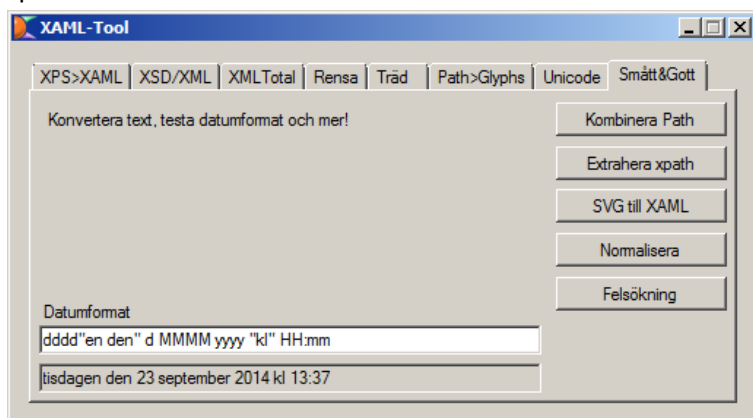
Sök

Det finns totalt 65535 förskjutningar att gå igenom. Om du inte hittar rätt förskjutning från början kan det bli svårt. Använd då denna funktion så kommer XAML-Tool att försöka hitta rätt åt dig. XAML-Tool kommer då att basera sökningen på hur många "vanliga" tecken som förekommer per förskjutning. Om du inte är nöjd med sökningen så kan du prova kryssa i/ur "Invertera" och försöka igen så kanske det går bättre.

Klicka sen på "Konvertera" för att konvertera och spara filen.

Smått&Gott

Här finns lite småfunktioner som kan användas till lite diverse, ofta inte direkt relaterat till någon speciell fil.



Datumformat

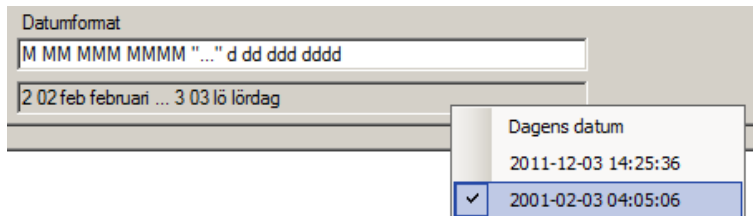
En smidig låda för att snabbt utforma en Date Time String (som formaterar datumformat). Någon dokumentation på Date Time String skriver jag inte här. Se istället <http://msdn.microsoft.com/en-us/library/8kb3ddd4.aspx>.

Skriv in tecken i lådan så kommer lådan under uppdateras med resultat, baserat på dagens datum.

Om du vill ha ett statiskt datum, [högerklicka](#) på resultatlådan och välj något av de andra alternativen.

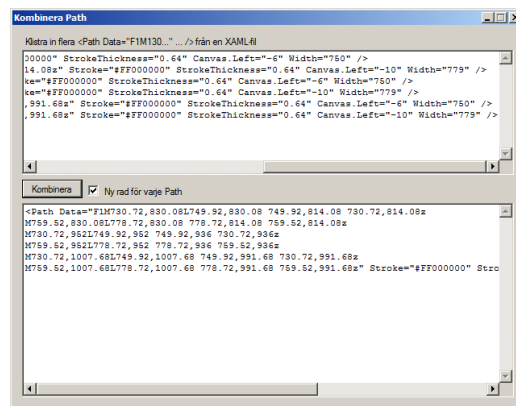
2011-12-03 14:25:36: Detta datum är utformat för att olika datumdelar ska vara unika, så att man kan skilja på dag och år i resultatet.

2001-02-03 04:05:06: Alla delar börjar på 0. Detta för att man ska se skillnad på t.ex. H (4) och HH (04).



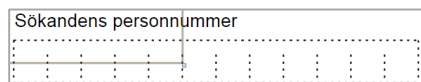
Kombinera Path

Ibland förekommer massa små Paths för samma "figur", t.ex. en punktruta. Det innebär att ungefär 33 bytes (+indentering) per Path skrivs i onödan. Dessutom är det jobbigt att hantera. Med denna funktion kan du enkelt sammansmälta alla dessa Pathar till en enda! Klicka på knappen "Kombinera Path". Nu får du upp ett nytt fönster med två textlådor. Lite exempelkod finns redan där. Den kan du ta bort. Klistra nu in ett antal Pathar från en XAML-fil i översta lådan. Tryck på "Kombinera". Resultatet presenteras i nedre lådan. Kopiera det och klistra in i din XAML-fil.



Om kryssrutan "Ny rad för varje Path" är ikryssad kommer Path-Datan skilja sig med en ny rad. Detta för att inte skapa förbaskat långa rader.

OBS: En Path kan endast använda en färg. Endast färgen från första Pathen i koden du klistrar in kommer att användas! Detta gäller även alla andra attribut, som *Stroke*, *Canvas.Left* och *Style*.

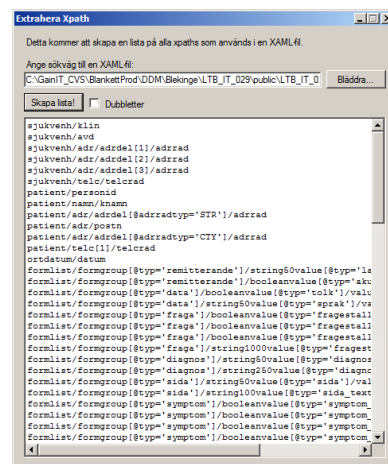


Exempel på när denna funktion är användbar. Varje punkt är en Path.

Extrahera xpath

Om du vill ha en lista på alla xpath:s som förekommer i en XAML-fil är det smidigt att använda denna funktion. Klicka på knappen "Extrahera xpath" så får du upp ett nytt fönster där du kan ange sökväg till en XAML-fil. När du gjort det kan du klicka på "Skapa lista!" så kommer den stora rutan fyllas med XAML-filens alla xpath:s. Om du vill ha unika xpath:s ser du till att "Dubletter" är urkryssad, annars kommer samma xpath förekomma fler gånger.

Listan är "sorterad" efter den ordning xpath:arna förekommer i XAML-filen.



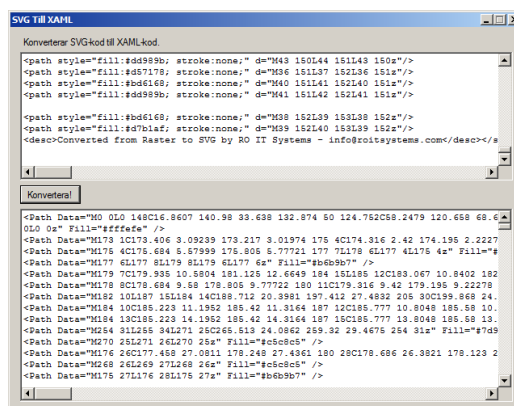
Konvertera SVG till XAML

Om du vill ha en vektoriserad bild i din blankett måste du först konvertera en rasterbild till vektor med hjälp av t.ex. [RO IT](#), men koden man får ut där går inte klistra in direkt i XAML-filen. Den måste omvandlas först! Detta gör du smidigast med detta verktyg.

Öppna din SVG-fil i en webbläsare och visa källa. Kopiera all text. Gå sedan till XAML-Tool och klicka först på "SVG till XAML" för att få upp fönstret. Klistra sedan in innehållet i SVG-filen i översta rutan, och klicka sen på "Konvertera!".

Nu kommer koden göras om så att du kan använda den i din XAML-fil. Resultatet hamnar i den undre rutan. Nu kan du kopiera texten och klistra in i din XAML-fil.

Sedan får du såklart använda Canvas och RenderTransform och så för att positionera, men det hör inte hit.



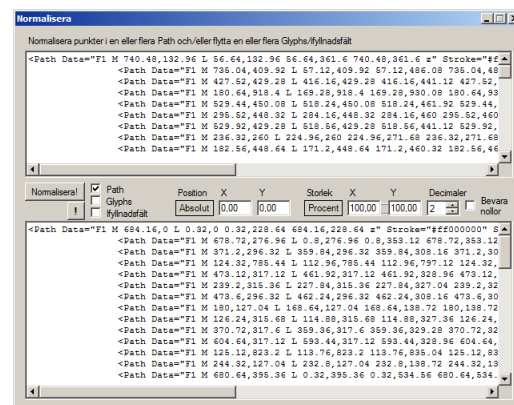
Normalisera

Denna funktion kan omvandla punkterna i Path:ar samt flytta på Glyphs och ifyllnadsfält på samma gång. Detta kan vara användbart särskilt om t.ex. en blankett fått ny layout, där halva sidan skjutits ner en aning.

Kopiera kod från din XAML-fil och klistra in i den övre rutan.

Det går jättebra att ha flera Path:ar och/eller

Glyphs/ifyllnadsfält på samma gång. Alla punkter i alla Path:ar räknas med i samma transformering. Gör några inställningar i rutorna och klicka på "Normalisera!".



Path

Detta omvandlar själva punkterna i en eller flera Path:ar. Med Detta kan du göra så att minst en X-punkt är 0, och samma med Y.

Glyphs

Flyttar på en eller flera Glyphs. Angiven position kommer att läggas till på OriginX och OriginY.

FontRenderingEmSize kommer baseras på genomsnittet av Storlek X och Storlek Y.

Ifyllnadsfält

Flyttar på en eller flera vanliga ifyllnadsfält. Angiven position kommer att läggas till på Canvas.Left och Canvas.Top. Endast Width påverkas av storleksangivelsen.

Position

Det finns två typer av position; Absolut och Relativ.

Absolut innebär att den vänstraste punkten i bilden kommer få den X-position du angivit i X, och den översta punkten kommer den Y-position du angivit i Y. Resten av punkterna kommer följa därefter.

Relativ position innebär att hela bilden förflyttas med angivna koordinater.

OBS: Absolut position fungerar bara på Path.

Position	X	Y
Relativ	0,00	0,00

Här kommer ingenting hända med positionen.

Storlek

Detta ändrar bildens storlek på två olika sätt:

Procent kommer ändra storleken enligt procent. Med 100 % händer ingenting, då bilden redan är 100 %. Alltså är 200 % dubbelt så mycket, och 50 % hälften.

För att spegelvända en bild kan man alltså ange -100 % på valfri axel.

Enheter ändrar storleken så att den blir exakt angivna enheter (samma enhet som vid förflyttning). Om 100 är angivet på X-axeln kommer skillnaden mellan den vänstraste punkten och högraste punkten vara 100.

Samma sak gäller översta och understa, dock då en helt annan axel.

För att omskala endast en axel och låta den andra följa med (utan att ändra ratio) kan man kryssa i rutan mellan X- och Y-rutorna.

OBS: Enheter går bara använda vid konvertering av Path.

Storlek	X	Y
Enheter	200,00	200,00

Decimaler

Ange antal decimaler som kommer med i resultatet för varje punkt. Det blir lätt virrigt med många decimaler för enstaka punkter och sådan precision är ofta inte nödvändig.

Genom att kryssa i "**Bevara nollor**" så kommer det alltid vara exakt så många decimaler som du angett.

T.ex så kommer värdet "3.2" bli till "3.20" om du angett två decimaler. Om kryssrutan är urkryssad så kommer endast nödvändiga decimaler med (dock max så många du angett).



Felsökning

Här kan du utföra en automatisk felsökning för att hitta vanliga misstag som ofta ställer till problem på ett eller annat sätt.



T.ex. om man angett en MaxLength som är mer än vad datatypen klarar av att hålla, eller om man angett en xpath som inte finns i XML:et.

Fyll i sökväg till XAML genom att klicka på knappen

"Bläddra...". Om du gör det kommer övriga sökvägar fyllas i automatiskt, förutsatt att filerna finns med liknande namn. Det går även bra att fylla i dessa var för sig om man vill.

Sedan är det bara att klicka på "Sök efter fel!" så kommer XAML-Tool att försöka hitta fel i filerna.

Alla fel som hittas fylls på i listan längst ner. Programmet hittar även varningar, som kanske inte direkt är fel men som man ändå bör se över.

Fel markeras med , och varningar markeras med .

