Version: 1.344
Datum: 2015-08-06

evry.com

Systemdokumentation

Systemdokumentation
EyeDoc™ Forms

Versionshistorik

Ändring nr.	Ändring datum	Förändringar	Reviderad av
1.9	2014-08-07	Hämtat in material till ny mall.	Erik Bunnstad
1.10	2014-08-08	Uppdateringar för ED 3.4.1, 3.4.3.	Erik Bunnstad
1.335	2014-12-19	Uppdaterat för ytterligare ändringar i ED 3.4 till 3.4.4.	Erik Bunnstad
1.339	2015-05-05	Uppdateringar för EyeDoc 3.5	Erik Bunnstad
1.344	2015-08-06	Kompletteringar av tidigare saknade delar.	Erik Bunnstad

Innehållsförteckning

1 Inledning

Forms är det huvudsakliga grafiska blankettverktyget i EyeDoc och används för visning och ifyllnad av blanketter. All användning av EyeDoc Forms är baserat på blanketterna.
När ett ärende har skapat som innehåller en digital EyeDoc blankett är det Forms som användaren kommer vara inne i varje gång som innehållet i blanketten ska visas eller redigeras.

Eftersom EyeDoc Forms är direkt beroende av en blankett för att kunna användas och styra vilka funktioner som är aktiva så beskriver detta dokument både tekniska lösningar i funktionerna som finns i Forms men också hur man använder sig av dessa i en blankett.

2 Systemdefinition

EyeDoc™ Forms är en klientapplikation som huvudsakligen publiceras som en XAML Browser Application (XBAP). När användaren blir länkad till Forms från EyeDoc Kuvert så kommer den senaste versionen som finns på servern laddas ner och installeras med ClickOnce och klienten uppdateras om applikationen saknas eller är av en tidigare version.

XBAP inför vissa begränsningar i vad Forms för göra då den inbyggda säkerhetsuppsättningen för XBAP förhindrar åtkomst av vissa funktioner utanför den egna applikationen.

XBAP och ClickOnce medför också att applikationen automatiskt installeras en gång per användare. Säkerhetsbegränsningarna medför att webbsiten som applikationen laddas ner ifrån måste ha korrekta säkerhetsinställningar från klientens internetinställningar.

EyeDoc Forms kan också publiceras som en EXE, men kan inte köras direkt utan måste köras av ett annat program. Detta är en av huvudsyftet med EyeDoc Reader för att komma ifrån de

problem och begränsningar som XBAP innebär och med det minskade stödet som finns för XBAP både utifrån och från Microsoft själva.

3 Översikt och flödesbeskrivning

För information om generella flöden se externt dokument EyeDoc Allmänt.
För information om specifika flöden i Forms se respektive kapitel för sökt funktion.

3.1 Översikt

Referens: Se kapitel 3 Översikt och flödesbeskrivning.

3.2 Flödesbeskrivning

Referens: Se kapitel 3 Översikt och flödesbeskrivning.

4 Funktionalitet

4.1 Grundlayout

(EB, 2014-08-08, Forms v1.1.0.330)

Forms består i grunden av en huvudmeny överst med funktionsknappar och metadata om personer som ärende och blankett rör. Undre delen består av två st DockPanel med uppsättningar av flikar. Den vänstra delen innehåller flikar med information och ytterligare funktioner. Den högra delen innehåller blanketten, informationssidor och bilagor.

4.1.1 Wrapper

(EB, 2014-08-08, Forms v1.1.0.330)

Skapat för att kunna publicera innehållet i Forms till olika format, både XBAP och EXE. Denna del har aldrig använts och har idag övergetts och ersatts av EyeDoc Reader.

4.1.2 Huvudmeny

(EB, 2014-08-08, Forms v1.1.0.330)

Huvudmenyraden i Forms innehåller följande knappar och funktioner:

Namn och personnummer för den patient till vilken ärendet hör.
Backaknapp
Status, ikon med tillhörande beskrivning**.**

- Blankettens nuvarande händelse - Sätts utifrån vad som inträffar för tillfället i Forms av respektive funktion.
- Ärendets status - baseras på `enum EyeDocFP.WS_IBIS_Kuvert_Ifyllnad.icIBIS_Status`.

Funktionsknappar

- Spara - Sparar blanketten som arbetsdokument (status 10).
- Spara Klar - Sparar blanketten som klar (status 20).
- Sign - Hel och delsignering av blankett. **Referens**: Se kapitel 4.7.4 Helsignering och 4.7.3 Delsignering.
- SignLock - eMU signering av blankett. **Referens**: Se kapitel 4.7.2 SignLock.
- Lock - Låser och/eller arkiverar blanketten. **Referens**: Se kapitel 4.8 Lock.
- Skicka - Skicka blankettens data som XML.
- eBrev - Skicka blanketten som eBrev. **Referens**: Se kapitel 4.13 ePrint.
- Utskrift - Öppnar upp utskriftsdialog för utskrift av aktiv flik. **Referens**: Se kapitel 4.4 Utskrift.
- SendFK - Skicka läkarintyg till Försäkringskassan. **Referens**: Se kapitel 4.14 Skicka Intyg.
- Makulera - Makulerar ärendet och blanketten. **Referens**: Se kapitel 4.9 Makulera.
- Markera fält - Visar och gömmer bakgrundsfärg för ifyllnadsfält.
- ePen - Visa och göm penstreck. Används inte, utan är helt ersatt av ePen fliken. **Referens**: Se kapitel 4.12 ePen.
- Hjälp - Visar hjälp och programinformation. **Referens**: Se dokumentation för EyeDoc.About.

Inloggad användares namn och titel.
Ansvarig användares namn och titel.

4.1.3Vänsterflikar

(EB, 2014-08-08, Forms v1.1.0.330)

Det går att gömma alla dessa flikar helt genom att kommentera bort trädet i blankettens `StackPanel.Resources`:

```
<!--<XmlDataProvider x:Key="treeData" XPath="/Data">
    <x:XData...>
</XmlDataProvider>-->
```

Notera att i vissa fall kan de stå `myData` istället för `treeData`, där den senare är den nya korrekta benämningen.

4.1.3.1*Översikt*

(EB, 2014-08-08, Forms v1.1.0.330)

Översikten visare en översiktöver innehållet i blanketten med konfigurerbara rubriker och fält. Trädet som det normalt kallas fungerar också som navigation i blanketten då varje nod länkas till ett element i blanketten.

Referens: Se kapitel 4.10 Träd.

4.1.3.2*ICD-10*

(EB, 2014-08-08, Forms v1.1.0.330)

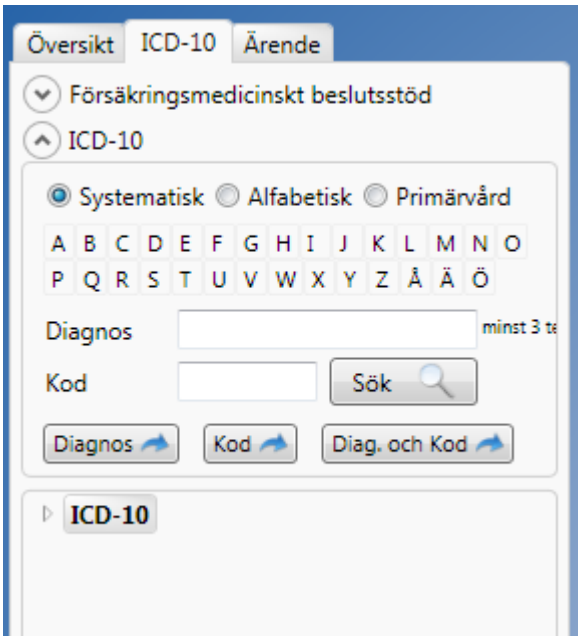
ICD-10 funktionen i EyeDoc Forms använder sig av de data som tillhandahålls av EyeDoc ICD10 (ICDCodeProvider) för att visa diagnoskoder grafiskt i olika vyer för användarna.

4.1.3.2.1 Layout

För ICD10SE finns det 3 vyer:

- Systematisk - Träd vy med de vanliga ICD10SE koderna
- Alfabetisk och Sök - Listningsvy av matchande koder i lista
- Primärvård - Träd vy med ICD10SE koder som har anpassats för primärvården

Oberoende vilken `RadioButton` som är vald (systematisk eller alfabetisk) så kan användaren söka, antigen på diagnos eller kod och sökresultat visas i en listningsvyn under. Om sökning görs när Primärvård är vald kommer sökningen göras på dessa koder annars är det träffar på de vanliga ICD10SE koderna som visas.



Observera att om man vill återgå till den alfabetiska vyn efter en sökning måste man klicka på någon bokstav för att få tillbaka vyn, eftersom alla vyer dela ett och samma visningsvyn.

Knapparna under sökfunktionen, kopiera över det information användare vill kopiera. Först måste användare välja vilken text fält information ska hamna (sist valda fält) sen klickar användare på någon av de tre knappar för att kopiera över information till fältet.

4.1.3.2.2 Funktion

All data som visas hämtas från EyeDoc ICDWebService och tjänsten ICDCodeProvider. Denna tjänst levererar ut de koder som visas i ett Xml format som sedan läses upp i EyeDoc Forms och fylls på i aktiv vy i ICD-10 fliken.

Referens: Se dokumentation för EyeDoc ICD10 eller servicekontraktet IProxyICDCodeProvider.cs.

I trädvyerna hämtas varje gren on-demand, vilket betyder att för varje ny gren som klickas upp så hämtas de undergrenar eller noder som finns under denna. Detta enligt den funktion ICDCodeProvider tillhandahåller och det är utformat på detta sätt för att begränsa den mängd data som skickas då antalet koder kan vara och är stort. De koder som hämtas lagras dock i klientprogrammet efter att de hämtats en gång, till dess att programmet stängs. De två träden för den Systematiska- och Primärvårdsuppsättningarna finns som två separata träd.

Sökfunktionen samt bokstavsknapparna gör ett nytt anrop och hämtar ny data för varje klick.

Knapparna för att fylla i Diagnos och eller Kod i ett fält lägger till texten, för den markerade koden från aktiv lista med koder, till det senaste aktiva ifyllnadsfältet i blanketten (det som senast haft focus). Funktionen stödjer enbart `TextBox`, är senast aktivt fält av en annan typ kommer inget att hända.

4.1.3.2.3 Utbyggnadsmöjligheter

Blanketten har idag ingen direkt påverkan på ICD10. Det har dock funnits tankar och intresse på att kunna definiera fält i en blankett dit kod, text, eller båda delar hämtas direkt in till dessa fält. Men man vill fortfarande ha kvar friheten att kunna hämta till vilket fält som helst.

I dagsläget används fasta koduppsättningsnamn, men det skulle vara möjligt att ändra detta i gränssnittet och lägga till möjligheter i webbtjänsten att hämta ut andra typer av koder förutom ICD10.

4.1.3.2.4 Försäkringsmedicinskt beslutsstöd

4.1.3.2.4.1 Layout

När användare söker på beslutstödet så länkas de ut till Socialstyrelsen försäkringsmedicinskt beslutstödshemsida. Sökfunktionen är en fritext sökning där användare kan mata text eller koder.

Observera att om till Socialstyrelsen försäkringsmedicinskt beslutstödssidan ändras sökväg så måste man kompilera om XBAP med den nya sökväg och uppdatera publiceringen.

Om användare har fyllt i fri text fältet så kommer det information också över till socialstyrelsen söksidan.

4.1.3.2.4.2 Funktion

I xbap.config.xml finns konfiguration för SocialStyrelseDecisionSupportUrl, den bas-url som går till Socialstyrelsens söksida för beslutsstöd på diagnoskoder.
När användaren sedan klickar på länken så läggs den, i tillhörande fält, skrivna texten till den konfigurerade url:en och öppnas i ett nytt webbläsarfönster.

4.1.3.3*Ärende*

4.1.3.3.1 Ärendeinformation

Visar grundläggande metadata om ärendet, blankettens namn och artikelnummer, antal versioner.

4.1.3.3.2 Inställningar

Fliken inställningar innehåller inställningar för ärendet.

4.1.3.3.2.1 Samtycke

(EB, 2014-08-08, Forms v1.1.0.330)

Här kan användaren ange om samtycke finns från patienten att skicka blanketten elektroniskt till en extern part, t.ex. som läkarintyget FK7263 skickas till Försäkringskassan.

Samtycke hanteras av EyeDoc Messenger och när man kryssar i eller ur skickas ett anrop till Messenger med uppgift om samtycke ska läggas till eller tas bort från ärendet. Messenger i sin tur sätter bitstatusar i Kuvertet för ärendet och den uppgiften används för att kryssa i eller ur rutan och skriva ut vad samtycket är när en sparad blankett öppnas.

De statusar som kan sättas för samtycke som består av två bitstatusar är om frågan har ställts och därefter om samtycke finns eller inte (Ej satt/Ja/Nej). Det går att sätta alla tre i anropet, men i inställningspanelen går det bara välja Ja eller Nej genom att kryssa i eller ur tillhörande ruta, det går inte att komma tillbaka till läget Ej satt (att frågan inte har ställts till patienten).

Eftersom ärendet måste finnas i både Kuvertet och Messenger för att kunna sätta samtycke finns funktion i Forms för att ändå kunna kryssa i rutan innan första sparningen har gjorts. Informationen om att samtycke ska sättas lagras då i en variabel i Forms som kommer göra att efter lyckad sparning kommer en timer startas som efter två sekunder försöker sätta samtycket via Messenger, ytterligare ett försök görs efter tre sekunder och därefter visas ett meddelande för användaren om båda försöken skulle misslyckas. Denna tidsfördröjning

finns för att statusmeddelande ska hinna fram till Messenger för att skapa ärendet där, innan dess kommer samtycke inte kunna sättas då Messenger inte känner till ärendet.

4.1.3.3.3 Versioner

(EB, 2014-08-08, Forms v1.1.0.330)

Listar de versioner som finns för ärendet, vilket nummer versionen har, dess status samt dess datum. Dubbelklickar man på en version kommer den att öppnas i versionsfliken.

4.1.3.3.4 Historik

(EB, 2014-08-08, Forms v1.1.0.330)

Visar historik om händelser som har inträffat rörande ärendet, detta kan inkluderar bland annat alla statusförändringar som inträffat. Här visas datum då händelsen inträffade, vem som vara inloggad samt ansvarig och vilken händelse det var.

4.1.3.4*Bilagor*

(EB, 2014-08-08, Forms v1.1.0.330)

Visar en lista över kopplade bilagor till blanketten samt en lista på tillgängliga bilagor som kan kopplas till blanketten. Denna flik visas enbart om blanketten har stöd för bilagor.
Referens: Se kapitel 4.15 Bilagor.

4.1.3.5*ePen*

(EB, 2014-08-08, Forms v1.1.0.330)

Fliken ePen (också känt som Digital Penna och Papper, DPoP) innehåller inställningar för, funktioner och data som kommer från en digital penna. Denna flik visas enbart om blanketten har stöd för att ta emot data från en digital penna.
Referens: Se kapitel 4.12 ePen.

4.1.3.6*Fråga/Svar*

(EB, 2015-08-05, Forms v1.1.0.344)

Fråga/Svar visar en XAML kontroll som listar och låter användaren välja dialoger att läsa, svara på inkomna frågor och skicka nya frågor. Fråga/Svar visas enbart när blanketten är skickad.

För att aktivera Fråga/Svar används inställningar i RegisterMedicalCertificate. **Referens:** Se kapitel 4.5.6 RegisterMedicalCertificate.

4.1.4Bottenpanel

(EB, 2014-08-08, Forms v1.1.0.330)

Används inte längre, har tidigare använts för att visa data från en digital Anoto penna, är ersatt av ePen fliken.

4.1.5 Huvudflikar

4.1.5.1 *Formulär*

(EB, 2014-08-08, Forms v1.1.0.330)

Innehåller huvudblanketten för ärendet. Denna flik visas alltid och läses in från blankettens XAML fil.

4.1.5.2 *Information*

(EB, 2014-08-08, Forms v1.1.0.330)

En extra flik där specifika sidor enbart innehållande information tillhörande blanketten, t.ex. informationssidor för hur man fyller i blanketten. Den kan innehålla ifyllnadsfält men har enbart begränsat skydd och validering så bör endast användas för låsta fält som får data från formulärfliken alternativt enklare värden som t.ex. utskriftsdag.
Referens: Se kapitel 4.24.2 Informationssidor.

4.1.5.3 *Versioner*

(EB, 2014-08-08, Forms v1.1.0.330)

I denna flik läses layout och ifyllnadsdata in för en specifik version av ärendet när denna version öppnas från versionsdelen i ärendefliken.

4.1.5.4 *Hjälp*

(EB, 2014-08-08, Forms v1.1.0.330)

Visar hjälp och programinformation. Det som visas är en direkt implementation av den information som lämnas ut av EyeDoc.About.
Referens: Se dokumentation för EyeDoc.About.

4.1.5.5 *Lista*

(EB, 2014-08-08, Forms v1.1.0.330)

Används inte, har aldrig slutförts, var tänkt att visa kuvertlistningen.

4.2 Öppna

(EB, 2014-08-08, Forms v1.1.0.330)

En blankett öppnas alltid genom att en session från Kuvertet används. Med hjälp av sessions-id:t så hämtar Forms blankettens XML- och XAML-filer från Innehållstjänsten och Arkivet via Kuvertet.

Om det är en ny blankett som öppnas är XML-filen en ny fil från Arkivet som eventuellt fyllts med ifyllnadsvärden eller innehåller värden från original XML:et i Arkivet. Är det en sparad blankett kommer den från Innehållstjänsten med det innehåll som har sparats där. Men i båda fallen så innehåller XML:et alla de uppgifter som kommer visas i fälten, som är kopplade till XML:et, i blanketten när den öppnas. **Referens**: Se kapitel 4.6 ElementTag.

Att anmärka här är att då att blankettens XML inte kommer att uppdateras från Arkivet efter att blanketten sparats, då skapas alltid XML:et från vad som finns sparat för ärendet i

Innehållstjänstens databas. Så eventuella uppdateringar i XML:et för samma version av blanketten i Arkivet kommer bara finnas om en ny blankett skapas. Detta kan vara viktigt att tänka på vid produktion av blanketter, läggs något till i XML:et som också används av ett fält i XAML måste en ny version av blanketten skapas annars kan det finnas tidigare sparade blanketter med fält vars värden inte kommer sparas.

4.2.1HandleError felkoder

(EB, 2014-08-08, Forms v1.1.0.330)

Om ett fel skulle inträffa vid öppnandet av en blankett i Forms visas en speciell felhanteringssida med följande felkoder.

Felkod	Beskrivning
101	Fel vid behandling av url
601	Fel vid hämta ärende
202	Fel vid hämta ärende, ingen XAML data
203	Fel vid hämta ärende, ingen XAML data, objekt null
204	Fel vid hämta ärende, ingen filtyp, objekt null
205	Fel vid hämta ärende, ingen XML data
206	Fel vid hämta ärende, ingen XML data, objekt null
207	Fel vid hämta ärende, inget resultat, objekt null
602	Fel vid signering av ärende
401	Fel vid signering av ärende, resultat objekt null
403	Fel vid signering av ärende, okänt fel
610	Fel vid uppläsning av XAML eller XAML för ePenna

4.3 Spara

(EB, 2014-08-08, Forms v1.1.0.330)

Allt som skrivs i fält i blanketten kommer att lagras i den XML-fil som Forms fick när blanketten öppnades. Sparning av dessa data görs sedan genom att detta XML skickas, via Kuvertet, till Innehållstjänsten som lagrar uppgifterna där i.

4.4 Utskrift

(EB, 2015-03-26, Forms v1.1.0.338)

Utskrift av Formulärfliken kan göras av det mesta innehåll. Border innehållande FixedPage och separata FixedPage skrivs ut genom att VisualBrush skapas av dem som placeras i en Rectangle och en ny FixedPage som skrivs ut. Eftersom ett FixedDocument behövs och de inte rakt av kan placeras i det då de redan har en förälder (nämligen ev. dess Border och sedan blankettens StackPanel), vilket VisualBrush går runt.

Annat innehåll som inte har en fast layout, som t.ex. kan hittas i en SVP skrivs ut genom att dela upp innehållet i mindre delar. Dessa element avbildas på samma sätt med en

VisualBrush som också placeras som fyllnad i en Rectangle, men som klipps till och positioneras efter varandra på nya FixedPage.

Utskrift av dynamiskt innehåll kommer idag alltid anpassas för utskrift på A4-format. Material av annat format kommer försöka anpassas till A4 vilket skulle kunna medföra innehåll som blir ihop tryckt, utdraget eller avklippt.

Det som också läggs till varje sida för dynamiskt innehåll är ett sidhuvud som kommer innehåller namn och nummer på blankett och patient samt sidnummer för varje sida. I dagsläget är det ett fast huvud som inte går att påverka med några inställningar.

Om blanketten innehåller dynamiskt innehåll (alltså inte enbart FixedPage) och minst en Expander kommer det i utskriftsdialogen visas en möjlighet att expandera samtliga Expander före utskriften. I annat fall kommer blanketten skrivas ut med Expander öppna eller stängda så som användaren har ställt dem under sessionen. Efter utskriften återställs varje Expander till det läge som de hade innan utskriften.

Nackdelen med att dela upp materialet är dock att det kan vara svårt att göra och kan förändra utseendet på innehållet. Bland annat kan marginaler och positioneringar försvinna/förändras och vissa delar kan bli utdragna eller försvinna om de inte har någon faktiskt storlek (t.ex. Canvas) och liknande. Det kan därför vara nödvändigt att justera blankettmallens layout eller utskriftsinställningar för att få till ett så bra resultat som möjligt.

Referens: Se kapitel 4.5.1.6 Utskrift av dynamiskt innehåll för mer information om inställningar som styr bl.a. uppdelning av dynamiskt innehåll.

Anledningen till att dela upp innehållet i mindre delar är att man annars stöter på kraftiga prestandaförsämringar vid stort innehåll då allt material i en VisualBrush troligen måste renderas vid utskriften även om det inte syns. Har man t.ex. 15 sidor material i samma VisualBrush som ska visas en femtondel på varje sida så finns det i praktiken 15*15 = 225 sidors material att rendera istället för 15.

Flikarna Hjälp och Versioner kan inte skrivas ut. Den senare för att validering saknas på den fliken vilket krävs av utskriftsfunktionen i vissa fall för att styra bl.a. utskriftsmarkeringar **Referens**: Se kapitel 4.5.1.3 Utskriftsmarkering/arbetskopia. Dessutom finns inte stöd för valideringsmarkeringar, sekretessutskrift m.m. Vill man skriva ut en tidigare version måste man idag därför öppna denna version i fullt läge från Kuvertlistningen.

Informationsfliken kan alltid skrivas ut utan begränsning.

Referens: Se kapitel 4.5.1 Utskriftsinställningar för tillgängliga inställningar för utskrift.

4.5 Formsettings

(EB, 2015-08-03, Forms v1.1.0.344)

Inställningar för blanketten görs i Formsettings och innehåller all blankettspecifik konfiguration.

```
<XmlDataProvider x:Key="formSettings" XPath="/Data">
  <x:XData>
    <Data xmlns="">
      <settingtype1/>
      <settingtype2/>
    </Data>
  </x:XData>
</XmlDataProvider>
```

Denna placeras på lämplig plats i blankettens globala resurser:

```
<StackPanel x:Name="form1"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
  <StackPanel.Resources>
    <ResourceDictionary>
      <styles/>
      <formsettings/>
      <treeview/>
      <signdata/>
      <checkdata/>
    </ResourceDictionary>
  </StackPanel.Resources>
</StackPanel>
```

Vissa funktioner aktiveras genom att lägga till deras inställningar i formsettings.
Alla inställningsparametrar under en funktion har standardvärden som är hårdkodade i Forms och används om funktionen är aktiv men inget värde har angetts eller om den specifika parametern har utelämnats.

Alla värden trimmas vilket betyder om man vill ha t.ex. ett mellanslag i början av en mening måste man använda sig av `<![CDATA[ ]]>`.

4.5.1Utskriftsinställningar

(EB, 2015-08-03, Forms v1.1.0.344)

För att kunna styra utskriftsfunktioner via blanketten ska följande läggas till i XAML-filen under formsettings.

Attributet `active` måste sättas till `1` för att utskriftsinställningarna ska aktiveras.

4.5.1.1*Grundinställningar*

Följande styr grundläggande inställningar för utskirftsfunktionen:

```
<printsetting active="1">
  <setting type="reqSaveBeforePrint">true</setting>
  <setting type="showPrintPopup">true</setting>
  <setting type="enableFormPrint">true</setting>
```

```
  <setting type="printAsDefaultOnPostSign">false</setting>
</printsetting>
```

Parameter	Värden	Beskrivning
enableFormPrint	**true**/false	Styr om blanketten ska kunna skrivas ut eller inte. Om detta parametern sätts till false så är utskriftsknappen inaktiverad i huvudmenyn.
reqSaveBeforePrint	**true**/false	Styr om blanketten behöver sparas före utskrift eller om det ska gå skriva ut blanketten utan att behöva spara den först.
showPrintPopup	**true**/false	Styr om dialog rutan ”Utskrift av formulär” ska komma upp eller inte. Den rutan innehåller bland annat fälten för att ange utskriftstyp, samt vem som begärt och vem som ska ha utskriften.
printAsDefaultOnPostSign	true/**false**	Styr om utskrift ska vara förvalt i efterbehandlingsdialogen för signering. **Referens:** Se 4.5.4 SignLock.

4.5.1.2 *Sekretessutskrift*

(EB, 2014-08-11, Forms v1.1.0.330)

En blankett kan skrivas ut som en sekretessutskrift, vilket möjliggör att vissa fält eller andra element göms vid val av den typen av utskrift.

Sekretessutskrift kräver att blanketten är sparad klar, antingen med eller utan valideringsvarningar (ofullständigt ifylld).

För att kunna styra utskriftsfunktioner för sekretessutskrift används följande:

```
<printsetting active="1">
  <setting type="enablePrivacyPrint">true</setting>
  <PrintPrivacyLabelText>Sekretessutskrift</PrintPrivacyLabelText>
  <PrivacyGroup name="hidden">
    <SourceXName>
      Diagnos;
      Diagnoskod;
      Upplysningar
    </SourceXName>
  </PrivacyGroup>
</printsetting>
```

Parameter	Värden	Beskrivning
enablePrivacyPrint	true/**false**	Om blanketten ska kunna skrivas ut som sekretessutskrift.
PrintPrivacyLabelText	text	Den text som ska användas för att beskriva sekretessutskriften, som visas i dialoger m.m. Hämtas från språkfilen om det inte är satt.

PrivacyGroup. SourceXName	x:Name	Namnen från x:Name attributet i XAML filen på de fält som ska vara sekretessbelagda.

Notera `name` attributet för `PrivacyGroup` måste vara `hidden`.

För att utskriftsdialogen ska fungera korrekt måste `enableMedicalCertificatePrint` aktiveras, **Referens:** Se kapitel 4.5.1.4 MedicalCertificatePrint.

4.5.1.3 *Utskriftsmarkering/arbetskopia*

(EB, 2014-08-11, Forms v1.1.0.330)

En blankett kan skrivas ut med en textmarkering när blanketten är under arbete. Detta är för att indikera att blanketten inte är ifylld klart.

För att kunna styra utskriftsfunktioner för arbetskopia används följande:

```
<printsetting active="1">
  <setting type="enablePrintMark">true</setting>
  <PrintMarkProperties>
    <SourceXName>PrintMarkElement1;PrintMarkElement2</SourceXName>
    <PrintMarkText>ARBETSKOPIA</PrintMarkText>
    <PrintMarkLabelText>Arbetskopia:</PrintMarkLabelText>
  </PrintMarkProperties>
</printsetting>
```

Parameter	Värden	Beskrivning
enablePrintMark	true/**false**	Om blanketten ska kunna skrivas ut som arbetskopia eller inte.
SourceXName	x:Name	Namnen från x:Name attributet i XAML filen på de fält ska visa markeringstexten. Dessa måste vara av typen Label.
PrintMarkText	Text	Den text som ska visas i dialoger, hämtas från språkfilen om det inte är satt.
PrintMarkLabelText	Text	Den text som ska visas på blanketten, hämtas från språkfilen om det inte är satt.

Följande är ett exempel på hur det element ska utformas som texten ska visas i. `Visibility` måste sättas till `Hidden` som standard för att den inte ska synas. Width måste vara så stor att den täcker texten man har valt i `PrintMarkText.` Obs texten som anges i elementet kommer att bytas ut mot `PrintMarkText` vid körning, men kan användas i produktionen för att förhandsgranska hur resultatet blir.

```
<Style x:Key="PrintMarkLabel" TargetType="Label">
  <Setter Property="FontSize" Value="32"/>
  <Setter Property="FontFamily" Value="Segoe UI"/>
  <Setter Property="FontWeight" Value="Bold"/>
  <Setter Property="Foreground" Value="Gray"/>
  <Setter Property="BorderBrush" Value="Gray"/>
  <Setter Property="BorderThickness" Value="0"/>
</Style>
<!--PrintMark - Arbetskopia-->
```

```
<Label x:Name="PrintMarkElement1" Visibility="Hidden" Style="{DynamicResource
PrintMarkLabel}">
  <Label.RenderTransform>
    <TransformGroup>
      <ScaleTransform ScaleX="3" ScaleY="2"/>
      <RotateTransform Angle="-45"/>
      <TranslateTransform X="120" Y="750"/>
    </TransformGroup>
  </Label.RenderTransform>
  ARBETSKOPIA
</Label>
```

För att utskriftsdialogen ska fungera korrekt måste `enableMedicalCertificatePrint` aktiveras, **Referens:** Se kapitel 4.5.1.4 MedicalCertificatePrint.

4.5.1.4*MedicalCertificatePrint*

(EB, 2014-08-11, Forms v1.1.0.330)

MedicalCertificatePrint används, till skillnad från vad namnet indikerar, för att ange att blanketten ska kunna skrivas ut fullt. Om inga andra utskriftsfunktioner är aktiverade går det alltid skriva ut blanketten normalt, men annars måste denna funktion aktiveras för att kunna skriva ut en full version av blanketten.

Denna typ av utskrift kräver att blanketten är korrekt validerad och spara som klar, men ger samma utskriftsresultat som en normal utskrift när inga andra utskriftsfunktioner är aktiverade. Det som skiljer i övrigt mot en normal utskrift är egentligen bara hur det loggas.

För att kunna styra utskriftsfunktioner för full utskrift används följande:

```
<printsetting active="1">
  <setting type="enableOriginalPrintLock">false</setting>
  <setting type="enableMedicalCertificatePrint">true</setting>
  <PrintMedicalCertificateText>Läkarintyg:</PrintMedicalCertificateText>
</printsetting>
```

Parameter	Värden	Beskrivning
enableMedicalCertificatePrint	true/**false**	Om blanketten ska kunna skrivas ut som läkarintyg eller inte.
PrintMedicalCertificateText	Text	Den text som ska visas i dialoger, t.ex. vid val av utskrift.
enableOriginalPrintLock	true/**false**	Styr om blanketten ska låsas vid MedicalCertificatePrint om blanketten inte är låst/signerad sedan tidigare. Default är false.

4.5.1.5*Ska till/PrintToText*

(EB, 2015-08-03, Forms v1.1.0.344)

PrintToText styr vilken text som ska stå i fältet för ”ska till” i utskriftsdialogen, alltså utskriftens syfte som loggas i utskriftsloggen. Anges inget här används istället texten från språkfilen.

För att kunna styra utskriftsfunktioner för full utskrift används följande:

```
<printsetting active="1">
  <setting type="enablePrintToText">true</setting>
  <PrintToText>Patient</PrintToText>
</printsetting>
```

Parameter	Värden	Beskrivning
enablePrintToText	true/**false**	Måste vara satt till true om texten ska hämtas från PrintToText.
PrintToText	Text	Den texts som ska visas i rutan, att lämna den tom ger samma effekt som att sätta enablePrintToText till false.

4.5.1.6 *Utskrift av dynamiskt innehåll*

(EB, 2015-03-26, Forms v1.1.0.338)

Dessa inställningar styr i första hand hur dynamiskt innehåll delas upp på sidor där det finns en prestandabegränsning som kräver uppdelningen från första början. Det finns även några inställningar för att styra hur innehållet och sidorna i sig ska se ut/formateras.

Förutom dessa inställningar kan blankettens layout i sig behöva justeras för att få till en bra utskrift. T.ex. så fungerar inte Margin korrekt på rootelement som placeras i en ViusalBrush och mindre element som inte har någon satt storlek kan ibland få en annan faktiskt storlek än vad man skulle kunna anta och när de hamnar i en ViusalBrush kan de då bete sig på oväntat sätt.

Följande inställning finns att göra för utskrift av dynamiskt innehåll:

```
<dynamicprint>
    <setting type="PrinterMargin">15</setting>
    <setting type="IndependentPrintableWidth">0.8</setting>
    <setting type="DuplicateContentHeight">10</setting>
    <setting type="ForceMergeHeight">1</setting>
    <setting type="ForceSplitHeight">5</setting>
    <setting type="StretchElementsWidth">0.8</setting>
    <setting type="ExpandExpanders">true</setting>
</dynamicprint>
```

Parameter	Värden	Beskrivning
PrinterMargin	Decimaltal 0-100	Sidmarginal i pixlar som ska läggas runt om innehållet på sidan. Standardvärde är 15 pixlar.

IndependentPrintableWidth	Decimaltal 0-2	Gränsvärde för om ett element ses som utskriftsbart för sig utan att behöva sin förälder. Värdet anges i sidbrädder där 1.0 är 100% av en sidas bredd (793.76 pixlar för A4). Element smalare än värdet kommer försöka skrivas ut med sin förälder och värden bredare försöks skrivas ut för sig utan föräldern. ForceMergeHeight och ForceSplitHeight kan dock göra att detta frångås. Standardvärde är 0.8 och gäller då alltså element större än 80% av en sidas bredd.
DuplicateContentHeight	Decimaltal 0-100	Antal pixlar av ett element som kommer upprepas på nästa sida om elementet inte får plats på föregående sida utan måste delas upp. Detta används för att göra det lättare att läsa text i de fall den klipps av med en sidbrytning rakt igenom texten. Standardvärde är 10 pixlar, om 0 anges kommer inget upprepas utan elementet fortsätter direkt på nästa sida där den blev avklippt på den förra sidan.
ForceMergeHeight	Decimaltal 0-100	Gränsvärde för element som kommer försöka skrivas ut med sin förälder även om det är tillräckligt brett enligt IndependentPrintableWidth för att skrivas ut för sig. Värdet anges i sidhöjder där 1.0 är 100% av en sidas höjd (1122.56 pixlar för A4). Används för att inte dela upp små element i onödan, vilket kan orsaka

		förändringar i elementets utseend up utskriften, då det ändå inte kommer göra någon märkbar prestandaskillnad. Om elementets förälder däremot är större än värdet för ForceSplitHeight kommer det går före och elementet skrivs ut för sig utan föräldern ändå. Standardvärde är 1.0 och gäller då alltså element mindre än 100% av en sidas höjd.
ForceSplitHeight	Decimaltal 0-100	Gränsvärde för element vars barn alltid kommer skrivas ut för sig även om IndependentPrintableWidth och/eller ForceMergeHeight anser motsatsen. Värdet anges i sidhöjder där 1.0 är 100% av en sidas höjd (1122.56 pixlar för A4). Används för att undvika prestandaproblem som uppkommer om stora element inte delas upp då det kommer innebära att innehållet renderingsmässig upprepas över flera sidor, även om bara en del av innehållet är synligt på varje sida. Det skapar annars en exponentiell prestandaåtgång, jämför t.ex. för 20 sidor: 4*(5^2) = 100 vs 1*(20^2) = 400. Standardvärde är 5.0 och gäller då alltså element större än 500% av en sidas höjd.
StretchElementsWidth	Decimaltal 0-2	Gränsvärde för element som kommer att sträckas ut i x-led för att fylla sidan även om de i verkligheten är mindre. Värdet anges i sidbrädder där 1.0 är 100% av en sidas bredd (793.76 pixlar för A4).

		Används för att strecka ut barnelement som normalt har marginaler mot en förälder som försvinner om de skrivs ut separat enligt övriga inställningar. Värde bör därför inte sättas för nära 0. Notera att skalningen enbart görs i x-led och inte y-led, utan element kommer då streckas eller tryckas ihop. Element som är bredare än sidan kommer tryckas ihop om de är större än sidan och större än angivet värde. Standardvärde är 0.8 och gäller då alltså element större än 80% av en sidas bredd.
ExpandExpanders	true/false	Anger om alternativet att expandera Expander Standardvärde är true.

4.5.2 Math

Referens: Se kapitel 4.18 Math.

4.5.3 Validation

(EB, 2014-08-12, Forms v1.1.0.331)

För att kunna styra inställningar för valideringsfunktioner från blanketten ska följande läggas till i XAML-filen under den befintliga formsettings:

```
<formsettings settingversion="1" formversion="1">
   <Validation active="1">
      <Version>2.0</Version>
   </Validation>
</formsettings>
```

Attributen active i `Validation` elementet måste sättas till ”1” för att valideringsfunktioner ska aktiveras i Forms.

`Version` styr vilken version som ska användas. ”`1.0`” är den gamla typen som inte stödjer de utökade valideringsfunktionerna, så ”`2.0`” ska alltid användas för att alla funktioner ska fungera.

Referens: För mer information om generell funktion och specifika valideringsfunktioner se kapitel 4.19 Validering.

4.5.3.1 *Valideringsdialoger*

(EB, 2014-12-19, Forms v1.1.0.335)

Det går i via blanketten styra vilka valideringsdialoger som ska visas samt vilka valideringsnivåer som krävs för en viss funktion. Detta görs med en `dialogs` nod som placeras direkt i `Validation` noden.

I följande exempel visas inställningar för att visa en dialog när blanketten öppnas:

```
<formsettings settingversion="1" formversion="1">
  <Validation active="1">
    <Version>2.0</Version>
    <dialogs>
      <function name="Open" block="Disabled" error="Disabled" warning="Disabled"
info="Important" specifictype="Error" specificdetails="false">
        <dialog type="Error">
          <caption>Reservnummer får inte användas!</caption>
          <messages>
            <message>"FK7263 - Läkarintyg" är inte tänkt att användas för
reservnummer.</message>
            <message>Intyget kommer inte kunna skickas elektroniskt till
Försäkringskassan!</message>
          </messages>
        </dialog>
        <dialog type="Info">
          <caption>Ofullständigt ifylld!</caption>
          <messages>
            <message>Blanketten som du precis öppnade innehåller ofullständigt ifyllda
uppgifter.</message>
          </messages>
        </dialog>
      </function>
    </dialogs>
  </Validation>
</formsettings>
```

4.5.3.2 *Inställningsbeskrivning*

(EB, 2014-08-12, Forms v1.1.0.331)

Följande beskriver de parametrar som kan ställas in i XML-delen för valideringsdialogerna:

Parameter	Godkända värden	Beskrivning
function.name	**Referens:** Se kaptiel 4.19.4 Funktionstyper (Enum.Parse)	Vilken funktion som inställningarna hör till.
function.block	**Referens:** Se kaptiel 4.19.1 Valideringsnivå (Enum.TryParse)	Vilken nivå som ska trigga dialogtypen Block.
function.error	**Referens:** Se kaptiel 4.19.1 Valideringsnivå (Enum.TryParse)	Vilken nivå som ska trigga dialogtypen Error.
function.warning	**Referens:** Se kaptiel 4.19.1 Valideringsnivå (Enum.TryParse)	Vilken nivå som ska trigga dialogtypen Warning.

function.info	**Referens:** Se kaptiel 4.19.1 Valideringsnivå (Enum.TryParse)	Vilken nivå som ska trigga dialogtypen Info.
function.specifictype	**Referens:** Se kaptiel 4.19.2 Dialogtyper (Enum.TryParse)	Vilken dialogtyp som ska användas för funktionsspecifika nivåer.
function.specificdetails	true/false (bool.TryParse)	Om lista med valideringsfel ska visas för funktionsspecifika nivåer.
dialog.type	**Referens:** Se kaptiel 4.19.2 Dialogtyper (Enum.Parse)	Vilken dialogtyp som inställningen hör till.
dialog.red	**Referens:** Se kaptiel 4.19.1 Valideringsnivå (Enum.TryParse)	Vilken nivå som ska innebära att fel listas som röda.
dialog.yellow	**Referens:** Se kaptiel 4.19.1 Valideringsnivå (Enum.TryParse)	Vilken nivå som ska innebära att fel listas som gula.
dialog.blue	**Referens:** Se kaptiel 4.19.1 Valideringsnivå (Enum.TryParse)	Vilken nivå som ska innebära att fel listas som blå.
caption	Fri text	Vilken text som ska visas i dialogens rubrik.
messages.message	Fri text	Vilka texter som ska i dialogen före (eventuella) valideringsfel, varje text visas på en egen rad.

Notera att funktionsspecifika nivåer inte bör användas som Valideringsnivå i attributen ovan, dessa ska enbart användas i valideringsregeln i tag på fältet. **Referens:** 4.19.5 Funktionsspecifika nivåer.

Notera också att speciell hantering finns för spara funktionerna, **Referens:** 4.19.4.1 BitStatus - Spara ofullständigt ifylld.

4.5.3.3 *Standardinställningar*

(EB, 2014-08-12, Forms v1.1.0.331)

Om en inställning för en specifik funktion/dialog inte anges i blanketten kommer Forms använda sina förbestämda standardinställningar. Följande listar alla funktioner som kan ha dialoger samt dess nivåer och texter som används som standard om inget annat anges:

```
<dialogs>
  <function name="Validate" block="Disabled" error="Disabled" warning="Disabled"
info="Fatal">
    <dialog type="Info" red="Fatal" yellow="Fatal" blue="Fatal">
      <caption>Ofullständigt ifylld!</caption>
      <messages>
```

```
        <message>Blanketten innehåller allvarliga fel som måste åtgärdas!</message>
      </messages>
    </dialog>
  </function>
  <function name="Open" block="Disabled" error="Disabled" warning="Disabled"
info="Important">
    <dialog type="Info" red="Important" yellow="Important" blue="Important">
      <caption>Ofullständigt ifylld!</caption>
      <messages>
        <message>Blanketten innehåller allvarliga fel som måste åtgärdas!</message>
      </messages>
    </dialog>
  </function>
  <function name="Save" block="Fatal" error="Disabled" warning="Critical"
info="Disabled">
    <dialog type="Block" red="Fatal" yellow="Important" blue="Important">
      <caption>Blanketten kan inte sparas!</caption>
      <messages>
        <message>Blanketten innehåller allvarliga fel som måste åtgärdas!</message>
        <message>Blanketten kan inte sparas!</message>
      </messages>
    </dialog>
    <dialog type="Warning" red="Fatal" yellow="Important" blue="Important">
      <caption>Spara blankett?</caption>
      <messages>
        <message>Vissa uppgifter saknas eller är ofullständigt ifyllda.</message>
        <message>Vill du spara ändå?</message>
      </messages>
    </dialog>
  </function>
  <function name="Done" block="Fatal" error="Mandatory" warning="Recommended"
info="Disabled">
    <dialog type="Block" red="Fatal" yellow="Important" blue="Important">
      <caption>Blanketten kan inte sparas!</caption>
      <messages>
        <message>Blanketten innehåller allvarliga fel som måste åtgärdas!</message>
        <message>Blanketten kan inte sparas!</message>
      </messages>
    </dialog>
    <dialog type="Error" red="Mandatory" yellow="Warning" blue="Recommended">
      <caption>Spara som arbetsdokument?</caption>
      <messages>
        <message>Viktiga uppgifter saknas eller är ofullständigt ifyllda.</message>
        <message>Blanketten kan endast sparas som arbetsdokument.</message>
        <message>Vill du spara?</message>
      </messages>
    </dialog>
    <dialog type="Warning" red="Mandatory" yellow="Warning" blue="Recommended">
      <caption>Spara blankett?</caption>
      <messages>
        <message>Vissa uppgifter saknas eller är ofullständigt ifyllda.</message>
        <message>Vill du spara ändå?</message>
      </messages>
    </dialog>
  </function>
  <function name="DonePreSign" block="Warning" error="Disabled" warning="Disabled"
info="Disabled">
    <dialog type="Block" red="Warning" yellow="Warning" blue="Recommended">
      <caption>Blanketten kan inte signeras!</caption>
```

```
      <messages>
        <message>Viktiga uppgifter saknas eller är ofullständigt ifyllda.</message>
        <message>Blanketten kan inte signeras!</message>
      </messages>
    </dialog>
  </function>
  <function name="SignLock" block="Warning" error="Disabled" warning="Recommended"
info="Disabled">
    <dialog type="Block" red="Warning" yellow="Warning" blue="Recommended">
      <caption>Blanketten kan inte signeras!</caption>
      <messages>
        <message>Viktiga uppgifter saknas eller är ofullständigt ifyllda.</message>
        <message>Blanketten kan inte signeras!</message>
      </messages>
    </dialog>
    <dialog type="Warning" red="Warning" yellow="Warning" blue="Recommended">
      <caption>Signera blankett?</caption>
      <messages>
        <message>Rekommenderade uppgifter saknas eller är ofullständigt
ifyllda.</message>
        <message>Vill du signera ändå?</message>
      </messages>
    </dialog>
  </function>
  <function name="SingleSign" block="Warning" error="Disabled" warning="Disabled"
info="Disabled">
    <dialog type="Block" red="Warning" yellow="Warning" blue="Recommended">
      <caption>Kan inte signera!</caption>
      <messages>
        <message>Viktiga uppgifter saknas eller är ofullständigt ifyllda.</message>
        <message>Signering kan inte utföras!</message>
      </messages>
    </dialog>
  </function>
</dialogs>
```

4.5.4 SignLock

(EB, 2015-08-03, Forms v1.1.0.344)

För att kunna styra låstning/signering via formblanketten ska följande läggas till i xaml-filen under den befintliga formsettings:

```
<SignLock active="1">
  <Version>1.0</Version>
  <setting type="eIDSignLockLevel">2</setting>
  <setting type="enablePreSignFunctions">true</setting>
  <setting type="enablePostSignFunctions">true</setting>
  <setting type="enablePostSignFunctionsPrint">true</setting>
  <setting type="titlesAllowedToSignEDForm">#lak#läk#läkare#överläkare#at
läkare#</setting>
  <setting type="showDocumentSignInfomation">true</setting>
  <setting type="showDoctorNameClarificationAndDate">true</setting>
  <setting type="allowSignForInvalidPatId">false</setting>
  <setting type="updatePatientIdXpath">patientED1/personnr</setting>
  <setting type="updatePatientNameXpath">patientED1/namn/knamn</setting>
  <setting type="PostSignDescriptionSentToPostBox">Intyget är nu signerat och sparat
på &quot;Mina Intyg&quot; och i Journalen.</setting>
  <setting type="PostSignAutoApprovalOnSend">true</setting>
```

```
  <setting type="PostSignSendHeader">Patienten samtycker och vill</setting>
  <setting type="PostSignPrintHeader">Patienten vill ha en utskrift av</setting>
  <setting type="PostSignSkipSendWarnings">true</setting>
  <DocumentSignInformation>
    <DocumentSignedStackPanel>DocumentSignedStackPanel</DocumentSignedStackPanel>
    <DocumentSignedByName>DocumentSignedByName</DocumentSignedByName>
    <DocumentSignedByTitleCode>DocumentSignedByTitleCode</DocumentSignedByTitleCode>
    <DocumentSignedTimeStamp>DocumentSignedTimeStamp</DocumentSignedTimeStamp>
    <DoctorNameClarification>AnsvarigNamn2</DoctorNameClarification>
    <SignatureDateTimeStamp>Signaturdatum</SignatureDateTimeStamp>
  </DocumentSignInformation>
  <setting type="showSignLockConfirmBox">true</setting>
  <SignLockConfirmDialog>
    <SignLockConfirmDialogMessage>Vill du skriva under och låsa
blanketten?</SignLockConfirmDialogMessage>
  </SignLockConfirmDialog>
</SignLock>
```

Parameter	Värden	Beskrivning
eIDSignLockLevel	0/1/2	Vilken typ av signering som blanketten ska användas. 0 = Ingen, kommer ge ett felmeddelande vid signering 1 = Hash/Session 2 = SITHS Om blanketten inte anger detta hämtas inställningen från xbap.config.xml.
enablePreSignFunctions	**true**/false	Om automatiska funktioner före signering ska användas (spara).
enablePostSignFunctions	**true**/false	Om automatiska funktioner efter signering ska användas (skicka och utskirft) i s.k. efterbehandlingsdialog.
enablePostSignFunctionsPrint	**true**/false	Om utskriftsfunktioner ska vara aktiva i efterbehandlingsdialogen. Kräver att `enablePostSignFunctions` är aktiverat.
titlesAllowedToSignEDForm	#titel#	Lista på de användartitlar som får signera blanketten. Dessa slås samman med de som finns angivna i xbap.config.xml, så finns en titel i någon eller båda av dem får denna signera.

showDocumentSignInfomation	**true**/false	Om signeringsinformation ska visas i blanketten efter signering i `DocumentSignInformation`.
showDoctorNameClarificationAndDate	true/**false**	Om extra namnförtydligande och signeringsdatum ska visas i signeringsinformationen på blanketten.
allowSignForInvalidPatId	**true**/false	Om signering ska tillåtas för patienter som inte har ett riktigt person- eller sammordningsnummer. Måste vara aktivt för att kunna signera med reservnummer eller liknande.
updatePatientIdXpath	xPath	xPath för att uppdatera patientens personnummer vid signering om det skilljer mellan systemet och blanketten. Utelämnas/lämnas som om den ej ska vara aktiv.
updatePatientNameXpath	xPath	xPath för att uppdatera patientens namn vid signering om det skilljer mellan systemet och blanketten. Utelämnas/lämnas som om den ej ska vara aktiv.
PostSignDescriptionSentToPostBox	Text	Text som ska visas i efterbehandlingsdialogen om blanketten har skickats till ExternalPostBox (t.ex. Intygstjänsten).
PostSignAutoApprovalOnSend	true/**false**	Om samtycke ska sättas automatiskt vid skicka från efterbehandligsdialogen utan att användaren får upp samtyckesdialogen.
PostSignSendHeader	Text	Rubrik som ska visas i efterbehandligsdialogen för att skicka.

PostSignPrintHeader	Text	Rubrik som ska visas i efterbehandligsdialogen för utskrift.
PostSignSkipSendWarnings	true/**false**	Om bekräftelsedialog för skicka i efterbehandligsdialogen ska hoppas över.
DocumentSignedStackPanel	x:Name	Namn på den `StackPanel` som ska visa signeringsinformation.
DocumentSignedByName	x:Name	Namn på den `TextBlock`/`TextBox` som ska visa namnet på den som signerade.
DocumentSignedByTitleCode	x:Name	Namn på den `TextBlock`/`TextBox` som ska visa titel på den som signerade.
DocumentSignedTimeStamp	x:Name	Namn på den `TextBlock`/`TextBox` som ska visa signeringens tidpunkt.
DoctorNameClarification	x:Name	Namn på den `TextBlock`/`TextBox` som ska visa namnförtydling på den som signerade.
SignatureDateTimeStamp	x:Name	Namn på den `TextBlock`/`TextBox` som ska visa signeringens tidpunkt.
showSignLockConfirmBox	true/false	Styr om bekräftelsedialog före signering ska visas.
SignLockConfirmDialogMessage	Text	Den text som ska står i bekräfteksedialogen före signering.

4.5.5 Lock

(EB, 2015-05-05, Forms v1.1.0.331)

Funktionen Lock används för att låsa en blankett utan att signera. Den kan också användas för att arkivera ärendet. Låsning och arkivering kan dock inte aktiveras separat utan antingen den ena, den andra eller båda.

Referens: Se kapitel 4.8 Lock för mer information.

Följande inställningar kan göras för funktionen:

Parameter	Värden	Beskrivning
LockEnable	**True**/false	Om funktionen ska vara aktiv. Standardvärde är true = aktiv.
LockButtonLockTooltip	Text	Tooltip för knappen vid låsning.
LockButtonCloseTooltip	Text	Tooltip för knappen vid arkivering.
LockButtonUnlockTooltip	Text	Tooltip för knappen vid upplåsning.
LockButtonOpenTooltip	Text	Tooltip för knappen vid öppning.
LockConfirm	**True**/false	Om dialog ska användas för att användaren ska bekräfta åtgärden. Standardvärde är true.
LockLock	**True**/false	Om åtgärden lås ska vara aktiv. Standardvärde är true, om inte SingLock är aktiverat, då gäller istället false.
LockLockConfirmCaption	Text	Dialogrubrik i bekräftelsedialog för åtgärden lås.
LockLockConfirmMessage	Text	Dialogtext i bekräftelsedialog för åtgärden lås.
LockClose	True/**false**	Om åtgärden arkivera ska vara aktiv. Standardvärde är false = inaktiv.
LockCloseConfirmCaption	Text	Dialogrubrik i bekräftelsedialog för åtgärden arkivera.
LockCloseConfirmMessage	Text	Dialogtext i bekräftelsedialog för åtgärden arkivera.
LockUnlock	True/**false**	Om åtgärden lås upp ska vara aktiv. Standardvärde är false = inaktiv.
LockUnlockConfirmCaption	Text	Dialogrubrik i bekräftelsedialog för åtgärden lås upp.
LockUnlockConfirmMessage	Text	Dialogtext i bekräftelsedialog för åtgärden lås upp.
LockOpen	True/**false**	Om åtgärden öppna (ta bort arkivering) ska vara aktiv. Standardvärde är false = inaktiv.
LockOpenConfirmCaption	Text	Dialogrubrik i bekräftelsedialog för åtgärden öppna.
LockOpenConfirmMessage	Text	Dialogtext i bekräftelsedialog för åtgärden öppna.

Notera att samtliga texter plockas från språkfilen istället om de saknas i blanketten och kan då utelämnas från inställningarna för att använda standardtexterna istället.

Exempel på hur inställningarna ser ut i XML-format:

```
<Lock active="1">
  <setting type="LockEnable">true</setting>
```

```
  <setting type="LockButtonLockTooltip">Lås och avsluta</setting>
  <setting type="LockButtonCloseTooltip">Arkivera</setting>
  <setting type="LockButtonUnlockTooltip">Lås upp</setting>
  <setting type="LockButtonOpenTooltip">Öppna</setting>
  <setting type="LockConfirm">true</setting>
  <setting type="LockLock">true</setting>
  <setting type="LockLockConfirmCaption">Bekräfta låsning</setting>
  <setting type="LockLockConfirmMessage">Vill du låsa blanketten?</setting>
  <setting type="LockClose">false</setting>
  <setting type="LockCloseConfirmCaption">Bekräfta arkivering</setting>
  <setting type="LockCloseConfirmMessage">Vill du arkivera blanketten?</setting>
  <setting type="LockUnlock">true</setting>
  <setting type="LockUnlockConfirmCaption">Bekräfta upplåsning</setting>
  <setting type="LockUnlockConfirmMessage">Vill du låsa upp blanketten?</setting>
  <setting type="LockOpen">true</setting>
  <setting type="LockOpenConfirmCaption">Bekräfta öppna</setting>
  <setting type="LockOpenConfirmMessage">Vill du öppna blanketten?</setting>
</Lock>
```

4.5.6 RegisterMedicalCertificate

(EB, 2015-08-03, Forms v1.1.0.344)

För att kunna skicka till Försäkringskassan via EyeDoc Forms/EDI RIV krävs det följande ska läggas till i xaml-filen under den befintliga formsettings:

```
<RegisterMedicalCertificate active="1">
  <Version>1.0</Version>
  <setting type="showApprovalDialog">true</setting>
  <setting type="showApprovalDialogIfNotAsked">false</setting>
  <ApprovalDialogMessage>
    Samtycker patienten till att ärendet skickas elektroniskt?
  </ApprovalDialogMessage>
  <StatusSentToExternalPart>
    <![CDATA[ ]]>till Försäkringskassan
  </StatusSentToExternalPart>
  <StatusSentToExternalPostBox>
    <![CDATA[ ]]>till Intygstjänsten
  </StatusSentToExternalPostBox>
  <setting type="SendToPostBoxOnSignature">true</setting>
  <setting type="SendToPostBoxFirst">true</setting>
  <setting type="SendToPostBoxRequresApproval">false</setting>
  <setting type="SendToExternalPartOnApproval">true</setting>
  <setting type="WarnBeforeSendToExternalPart">true</setting>
  <setting type="WarnBeforeSendToExternalPartText">Är du säker på att intyget ska
skickas till Försäkringskassan?</setting>
  <setting type="CancelFormConfirmForSentExternalPart">Makuleringsmeddelande
skickas{0} och i förekommande fall{1}.</setting>
  <setting type="CancelFormConfirmForSentPostBox">Makuleringsmeddelande skickas{0} och
i förekommande fall{1}.</setting>
  <ApplicationIds>
    <SendToExternalPostBox>11111111-1111-1111-1111-
111111111111</SendToExternalPostBox>
    <SendToExternalPart>44444444-4444-4444-4444-444444444444</SendToExternalPart>
    <SendQuestionMessage>22222222-2222-2222-2222-222222222222</SendQuestionMessage>
    <SendAnswerMessage>33333333-3333-3333-3333-333333333333</SendAnswerMessage>
  </ApplicationIds>
  <messengerdialogs enabled="true"/>
</RegisterMedicalCertificate>
```

Parameter	Värden	Beskrivning
showApprovalDialog	**true**/false	Om samtyckesdialog ska visas innan skickning. Om denna inaktiveras förutsätts att man får skicka utan samtycke.
showApprovalDialogIfNotAsked	**true**/false	Om samtyckesdialogen ska visas automatiskt vid spara eller i efterbehandlings-dialogen om frågan inte har ställts tidigare. Kräver att `showApprovalDialog` är aktiv.
ApprovalDialogMessage	Text	Det meddelande som ska visas i samtyckesdialogen.
StatusSentToExternalPart	Text	Tilläggstext till statusikonen i menyn vid Skicka till `ExternalPart`.
StatusSentToExternalPostBox	Text	Tilläggstext till statusikonen i menyn vid Skicka till `ExternalPostBox`.
SendToPostBoxOnSignature	true/**false**	Om blanketten ska skickas automatiskt till `ExternalPostBox` efter lyckad signering. Kräver att `SendToPostBoxFirst` är aktiv.
SendToPostBoxFirst	**true**/false	Om blanketten ska skickas till `ExternalPostBox` innan den skickas till `ExternalPart`.
SendToPostBoxRequresApproval	true/**false**	Om skicka till `ExternalPostBox` kräver samtycke. Kräver att `showApprovalDialog` är aktiv.
SendToExternalPartOnApproval	true/**false**	Om användaren ska få fråga om att skicka blanketten om den är signerad när samtycke sätts i blankett-inställningarna.

WarnBeforeSendToExternalPart	true/**false**	Om bekräftelsedialog ska visas för att skicka till `ExternalPart`.
WarnBeforeSendToExternalPartText	Text	Meddelandetext som ska visas i bekräftelsedialog för att skicka till `ExternalPart`.
CancelFormConfirmForSentExternalPart	Text	Eventuell extra meddelandetext för makuleringsdialogen om blanketten skickats till `ExternalPart`.
CancelFormConfirmForSentPostBox	Text	Eventuell extra meddelandetext för makuleringsdialogen om blanketten skickats till `ExternalPostBox`.
ApplicationIds	Guid	EyeDoc EDI ApplicationId som ska användas av blanketten för respektive funktion.
messengerdialogs	true/**false**	Om Fråga/Svar ska vara aktivt för blanketten när den har skickats. **Referens:** Se kapitel 4.1.3.6 Fråga/Svar.

4.5.7 EMUMessenger

(EB, 2015-08-03, Forms v1.1.0.344)

```
<EMUMessenger active="1">
  <Version>1.0</Version>
  <setting type="showCloseArchiveOption">false</setting>
  <CloseArchiveOptionLabel>Arkivera:</CloseArchiveOptionLabel>
</EMUMessenger>
```

Parameter	Värden	Beskrivning
showCloseArchiveOption	true/**false**	Om automatarkivering ska användas som arkiverar blanketten om man också väljer att slutföra utskrift via valet i efterbehandlingsdialogen.
CloseArchiveOptionLabel	Text	Den text som ska visas i efterbehandlingsdialogen för automatarkivering.

4.5.8 HighLightStyles

(EB, 2014-12-17, Forms v1.1.0.335)

HighLightStyles (även HighLightField, HighLightSetting) styr vilka stilar som ska gömmas genom att bytas ut till en alternativ stil med annat utseende. Detta används när man från menyn väljer att markera/avmarkera fält samt vid utskrift.

Detta sker genom att samtliga element i aktiv tab i Forms gås igenom rekursivt och stilen för samtliga element, som har en stil som ska bytas ut, ersätts med den alternativa stilen.

Det finns ett antal stilar som hanteras automatiskt av Forms som standard:

Måltyp	Stil	Utskriftsstil
TextBox	tbStyle	tbStyleT
CheckBox	cbStyle	cbStyleT
ComboBox	comboStyle	comboStyleT

Om andra stilar en de ovan angivna används för fält eller annat som ska ha ett annat utseende måste dessa stilar konfigureras i Formsettings under `highlightstyles/styleslist`.

Formsettings XML:

```
<highlightstyles active="1">
    <styleslist>
        tbBorderStyle:tbBorderStyleT;
        cbBorderStyle:cbBorderStyleT
    </styleslist>
    <refreshlist>
        canvasTbParentStyle;
        canvasCbbParentStyle;
        canvasCbParentStyle;
    </refreshlist>
</highlightstyles>
```

4.5.8.1 *Styleslist*

Under `highlightstyles/styleslist` ska de stilar som ska bytas ut vid gömmande av fält konfigureras.

Varje uppsättning av stilar ska separeras med semikolon (;) och bestå av namnet på de två stilar separerade med kolon (:). Den första stilen ska vara standardstilen och den andra ersättningsstilen.

4.5.8.2 *Refreshlist*

`highlightstyles/refreshlist` är inte implementerat.

Tanken är att det ska till en funktion som ska lägga in en speciell uppdatering på fält med avancerade stilar som kräver att kopplingen till stilen ställs om när blanketten har laddats för att de vid skapandet refererar till exempel barn element som inte än har skapats.

Se t.ex. ASVP_OMV där det finns stil som refererar till barn enligt:

```
<Setter Property="Width" Value="{Binding RelativeSource={RelativeSource
Self},Path=Children[0].ActualWidth}"/>
<Setter Property="Height" Value="{Binding RelativeSource={RelativeSource
Self},Path=Children[0].ActualHeight}"/>
```

Denna fungerar inte om stilen läggs till i XAML eftersom när föräldern som har stilen skapas så finns inte Children[0] eftersom det inte har skapats än. Om man däremot sätter om stilen så fungerar den som den ska (fungerar t.ex. om man lägger dem under styleslist, men då måste man manuellt klicka på markera/avmarkera fält).

4.5.9 ClassificationCodes

Referens: Se kapitel 4.20 Classification.

4.5.10 DataTransfer

Referens: Se kapitel 4.21 DataTransfer.

4.5.11WindowAreas

(EB, 2015-08-03, Forms v1.1.0.344)

WindowAreas tillåter att styra vilka fönster som ska vara aktiva, synas och ha fokus. Idag stödjs att sätta fokus när programmet laddas. I framtiden är det också tänkt att detta ska kunna styra var man placerar olika fönster.

Ett XML kan se ut enligt följande:

```
<windowareas active="1">
  <area name="Left">
    <windowfocus>
      <focus window="CaseInformation">
        <requires>...</requires>
      </focus>
      <focus window="ClassificationCodes">
        <requires>...</requires>
        <forbidden>...</forbidden>
      </focus>
      <focus window="MessengerDialogs">
        <conditions>
          <requires>...</requires>
        </conditions>
      </focus>
      <focus window="TreeOverview" />
    </windowfocus>
    <window name="CaseInformation" />
    <window name="TreeOverview" />
    <window name="DigitalPen" />
    <window name="MessengerDialogs">
      <enabled>
        <requires>...</requires>
        <forbidden>...</forbidden>
```

```
      </enabled>
      <visible>
        <requires>...</requires>
        <forbidden>...</forbidden>
      </visible>
    </window>
    <window name="ClassificationCodes" />
    <window name="Attachments" />
  </area>
  <area name="Main">
    <window name="FormContent" />
    <window name="FormInformation" />
    <window name="Version" />
    <window name="Help" />
  </area>
</windowareas>
```

Notera här elementen `area.windowfocus.focus`, `area.window.enabled` och `area.window.visible` samtliga består samtliga av `conditions`. Dess underelement `requires` och `forbidden` kan anges antingen inneslutna i en `conditions` nod eller separata direkt under respektive förälder. **Referens:** Se kapitel 4.5.12 Conditions för mer information.

Parameter	Värden	Beskrivning
area	windowfocus{0-1}, window{0-*}	Representerar en yta i ett program. Det är upp till programmet att tolka vilken yta detta är.
area@name	Left, Main	Namnet på ytan som ska konfigureras.
windowfocus	focus{0-*}	Styr vilket fönster i ytan som ska ha fokus.
focus	Conditions{0-1}	Villkorinställningar för att ge ett specifikt fönster fokus. **Referens:** Se kapitel 4.5.12 Conditions.
focus@window	Se lista på fönster nedan.	Namnet på det fönster som ska få fokus om villkoren är uppfyllda.
area/window	window{0-*}	Inställningar för ett specifikt fönster.
window@name	Se lista på fönster nedan.	Namn på det fönster som inställningarna tillhör.
window/enabled	Conditions{0-1}	Styr om fönstret ska vara aktivt eller låst. **Referens:** Se kapitel 4.5.12 Conditions.
window/visible	Conditions{0-1}	Styr om fönstret ska vara synligt eller inte. **Referens:** Se kapitel 4.5.12 Conditions.

Följande är de namngivna fönster som kan konfigureras:

WindowName	Beskrivning

CaseInformation	Ärendeinformation. **Referens:** Se kapitel 4.1.3.3 Ärende.
TreeOverview	Navigationsfönstret. **Referens:** Se kapitel 4.1.3.1 Översikt.
DigitalPen	DPoP. **Referens:** Se kapitel 4.1.3.5 ePen.
MessengerDialogs	Fråga/Svar. **Referens:** Se kapitel 4.1.3.6 Fråga/Svar.
ClassificationCodes	Kodverk/ICD10. **Referens:** Se kapitel 4.1.3.2 ICD-10.
Attachments	Bilagor. **Referens:** Se kapitel 4.1.3.4 Bilagor.
FormContent	Blankettinnehåll. **Referens:** Se kapitel 4.1.5.1 Formulär.
FormInformation	Blankettinformation. **Referens:** Se kapitel 4.1.5.2 Information.
Version	Blankettversion. **Referens:** Se kapitel 4.1.5.3 Versioner.
Help	Hjälpfliken. **Referens:** Se kapitel 4.1.5.4 Hjälp.

4.5.12 Conditions

(EB, 2015-08-03, Forms v1.1.0.344)

Conditions är en generell klass som är tänkt att kunna användas på olika platser i formsettings för att kontrollera om specifika villkor är uppföljda för att sedan utföra olika uppgifter beroende på dess resultat. Conditions används i första hand som en hjälpklass till andra klasser i Formsettings.

När Conditions används skickar den som anropar in ett antal parametrar, bland annat ärendets status. Dessa används sedan för att avgöra om de villkor som man satt upp är uppfyllda. Conditions kommer då att ge tillbaka detta resultat som den anropande klassen i sin tur kan använda för att avgöra vad som ska utföras.

4.5.12.1 *Requires och forbidden*

Conditions har två huvuddelar `requires` och `forbidden`. Noden `requires` definierar vilka underliggande villkor som måste vara uppfyllda för att ge tillbaka ett positivt resultat. Noden `forbidden` definierar vilka underliggande villkor som förhindrar ett positivt resultat.

Dessa stödjer att användas antingen under en förälder-nod `conditions` eller utan den direkt under den överliggande funktionens nod. Anledningen till det är helt enkelt för att man ska kunna minimera antalet XML-noder som behövs.

Med `conditions` nod:

```
<parentclassnode>
  <conditions>
    <requires>...</requires>
    <forbidden>...</forbidden>
  </conditions>
</parentclassnode>
```

Respektive utan:

```
<parentclassnode>
  <requires>...</requires>
  <forbidden>...</forbidden>
</parentclassnode>
```

4.5.12.2 *All och any*

För att kunna kontrollera att flera villkor är uppfyllda används sedan noderna `all` och `any`. Noden `all` säger alla villkor måste vara uppfyllda och `any` säger att något villkor (minst 1st) måste vara uppfyllt utan att alla behöver det.

Även här kan en omslutande nod utelämnas och villkor placeras direkt i `requires` eller `forbidden`. Om den utelämnas kommer villkoren tolkas som `all` så att:

```
<requires>
  <!--condition 1-->
  <!--condition 2-->
</requires>
```

Ger samma resultat som:

```
<requires>
  <all>
    <!--condition 1-->
    <!--condition 2-->
  </all>
</requires>
```

Vill man istället använda sig av `any`, så måste man ange det:

```
<requires>
  <any>
    <!--condition 1-->
    <!--condition 2-->
  </any>
</requires>
```

Man kan därefter kombinera dessa som man vill:

```
<requires>
  <all>
    <!--condition 1-->
    <!--condition 2-->
    <any>
      <!--condition 3-->
      <all>
        <!--condition 4-->
        <!--condition 5-->
```

```
      </all>
    </any>
  </all>
</requires>
```

Ovan motsvarar följande pseudo kod: 1 && 2 && (3 || (4 && 5)).

4.5.12.3 *Villkor*

Villkor motsvarar sedan de fördefinierade kontroller som kan göras mot de parametrar som skickas med från den anropande metoden.

Följande villkor finns definierade:

Villkor	Beskrivning
casestatus	Kontroll mot ärendets status. Per definition så kan ett ärende bara ha en status. **Referens:** Se dokumentation för DDM_Kuvert över vilka statusar som finns.
bitstatus	Kontroll mot ärendets bitstatus. **Referens:** Se dokumentation för DDM_Kuvert över vilka statusar som finns.

Också villkors värden kan slås samman med hjälp av semikolon (:) så att:

```
<any>
  <bitstatus values="64;128;256" />
  <casestatus values="10;20" />
</any>
```

Ger samma resultat som:

```
<any>
  <bitstatus values="64" />
  <bitstatus values="128" />
  <bitstatus values="256" />
  <casestatus values="10" />
  <casestatus values="20" />
</any>
```

4.5.12.4 *Exempel*

Signerad och skickad, men inte makulerad:

```
<conditions>
  <requires>
    <casestatus values="30" />
    <any>
      <bitstatus values="4096;65536" />
    </any>
  </requires>
  <forbidden>
    <bitstatus values="8" />
```

```
  </forbidden>
</conditions>
```

Signerad med någon typ av signering och skickad:

```
<requires>
  <casestatus values="30" />
  <any>
    <bitstatus values="64;128;256" />
  </any>
  <any>
    <bitstatus values="4096;65536" />
  </any>
</requires>
```

4.6 ElementTag

(EB, 2014-08-07, Forms v1.1.0.331)

För att ange referens, var värdet ska sparas, samt formaterings- och valideringsregler för ett fält används elementets Tag attribut. För att erhålla bakåtkompabilitet sätts en standard konverterare vid fel.

Syntaxen för Tag attributet är enligt:

```
<Tag>ElementName|XPath|Converter|ConverterParameter|Rule|ExtendedRule1|ExtendedRule2|E
xtendedRuleN</Tag>
```

ElementTag rensar inledande och avslutande blanksteg för samtliga parametrar, alltså i direkt kontakt med varje ”|” samt i början och slutet av tag attributet. Det går då att placera vissa parametrar på egna rader för att de ska bli lättare att läsa om de är för långa:

```
<Tag>
  ElementName|XPath|Converter|ConverterParameter|Rule
  |ValidationLevelA;ExtendedRule1;Param1:Param2:ParamN
  |ValidationLevelA;ExtendedRule2;Param1:ParamN
  |ValidationLevelB;ExtendedRuleN;Param1:Param2:Param3:ParamN
</Tag>
```

Dock får radbryt inte göras inom en parameter annat än mellan underparametrar i ExtendedRule. Normalt är det dock mest läsbart att inte radbryta i en ExtendedRule, utan ha alla dess parametrar på samma rad.

Namn	Beskrivning
ElementName	Ett visuellt namn på elementet som kan användas för att beskriva för användaren av blanketten vilket fält som refereras till, används bl.a. i valideringsfelmeddelanden.
XPath	Sökväg till den plats i XML:et där fältets värde ska sparas.
Converter	Styr vilket format som värdet i fältet ska konverteras till, vilket möjliggör t.ex. att d och t kan användas i datum respektive tidsfält.

ConverterParameter	Parametrar till konverteraren, som styr hur den ska konvertera värdet.
Rule	Regelverk som ska användas för att validera värdets format (och således datatyp).
ExtendedRule	Utökade valideringsregler som möjliggör mer avancerade valideringar. Flera av dessa regler kan användas samtidigt på ett och samma fält. **Referens:** Se kapitel 4.6.6 ExtendedRule.

Tabell: Förklaring av Tag attributet.

4.6.1ElementName

(EB, 2014-08-11, Forms v1.1.0.331)

ElementName är det namn på ett fält som kommer visas för användaren i meddelanden som refererar till det specifika fältet. Detta används bland annat för att visa valideringsfel för användaren i valideringsdialogerna, där det då skulle kunna stå t.ex. ”Fältet ’ElementName’ innehåller ett felaktigt datum.”.

Om ett ElementName inte anges kommer automatiskt fältets XAML namn användas.

4.6.2 XPath

(EB, 2014-08-11, Forms v1.1.0.331)

XPath är adresseringen till den plats i XML-filen där fältets värde kommer att sparas. När en blankett öppnas kommer XAML fältets värde kopplas med en two-way binding till den angivna noden i XML-filen. När värdet på fältet sedan ändras kommer XML-filen automatiskt uppdateras och tvärt om. En bieffekt av detta är att när XML-filen uppdateras så kommer samtliga fält i blanketten med en binding att uppdateras och tillhörande Converter kommer köras.

När uppdateringen från fältet sker styrs av fältets typ, t.ex. inträffar det när man lämnar fältet för TextBox, samtidigt som en ComboBox uppdateras under tiden man skriver i den och en CheckBox uppdaterar när man kryssar i eller ur den.

För mer info om hur XPath fungerar och exempel se t.ex.

1. http://www.w3schools.com/XPath/default.asp
2. http://msdn.microsoft.com/en-us/library/ms256086(v=vs.110).aspx

4.6.3 Converter

(EB, 2014-08-12, Forms v1.1.0.331)

Converter är den som konverterar värdet som skrivits i fältet till det som kommer sparas i XML-filen när man skriver i fältet och tillbaka från XML-filen till fältet när ändringar görs i XML-filen.

Flyttandet av värden fram och tillbaka görs automatiskt av den binding som skapas mellan fält och XML, Convertern är det tillägg till denna flytt som Forms gör. På så sätt kan Forms

manipulera värdet mellan de två platserna, t.ex. för att en kryssruta som är en boolean ska kunna sparas som värdet ”ja” i XML-filen med hjälp av stringEnumBooleanConverter.

Convertern kan också validera värdet enligt sitt format för att t.ex. med stringDateConverter skydda ett datumfält som har en xsDateRule från ett värde från XML-filen som inte är ett korrekt datum. Skulle värdet vara felaktigt skickas istället en tom sträng tillbaka i konverteringen av värdet. Detta fungerar också från fältet till XML-filen även om det är tänkt att det är Rule som ska hantera det för att få en markering på fältet för användarens skull.

Konverterare	Parameter	Fälttyper
BooleanConverter	0 | 1	CheckBox
stringBooleanVisibilityConverter	-	Border
stringCode128Converter	-	TextBox, TextBlock
stringEyeDocBarcodeConverter	enBarCode; enWight; enHight	TextBox, TextBlock
stringDateConverter	- | [dxx]	TextBox, TextBlock, Calendar, DatePicker
stringDoubleConverter	-	TextBox, TextBlock
stringEncodedConverter	-	TextBox, TextBlock, ComboBox, Calendar, DatePicker
stringEnumBooleanConverter	enumValue	Checkbox
stringInt32Converter	-	TextBox, TextBlock
stringRegExpConverter	RegExp pattern (används av xsRegExpRule)	TextBox, TextBlock
stringTimeConverter	- | [txx]	TextBox, TextBlock, Calendar, DatePicker
stringYearConverter	-	TextBox, TextBlock

Konverterare med parametrar och vilka fälttyper som de kan användas till. Notera att ovan tabell måste följas, det går inte fritt att kombinera konverterare och fälttyp utanför kombinationerna i tabellen.

4.6.3.1 *BooleanConverter*

(EB, 2014-08-11, Forms v1.1.0.331)

Konverterar boolean värdet (IsChecked) för ett kryss från en CheckBox till värdet “1” för ikryssad eller “null” för urkryssad (en tom sträng sparas vilket Innehållstjänsten tolkar som null).

Om värdet i XML-filen inte är något av ”1”, ”true” eller ”True” så kommer CheckBox:en vara ur-kryssad.

4.6.3.2 *stringBooleanVisibilityConverter*

Används inte längre då denna enbart kan användas för Border element, istället kan en BooleanToVisibilityConverter användas.

Referens: Se kapitel 4.11.2 Gömma/visa område efter kryssruta.

4.6.3.3 *stringCode128Converter*

(EB, 2014-08-11, Forms v1.1.0.331)

Konverterar värdet i XML-filen som är de tecken för respektive streckkombination som finns i streckkodstypsnittet till ett läsbart numeriskt värde.

Om man ska ha en streckkod med Code128 så måste dess Converter användas dessutom för att få en korrekt streckkod då denna lägger till start och slut på koden.
(`stringCode128Converter||xsCode128Rule`)

4.6.3.4 *stringEyeDocBarcodeConverter*

(ME, 2014-04-xx, Forms v1.1.0.326)

Om man ska ha en streckkod med Interleaved 2 of 5 eller Code 39 så måste stringEyeDocBarcodeConverter convertern användas för att få en korrekt streckkod då denna lägger till start och slut på koden, samt kodar strängen.
Fonten ”/Resources/Fonts/#EyeDocBarcode” skall användas.
stringEyeDocBarcodeConverter har en kopplad rule ”xsEyeDocBarcodeRule”, samt tre enum parametrar.

”enBarCode”, som anger vilken barcode som skall användas:

- Interleaved2of5
- Code39

”enWight”, som anger vilken modul/Ratio som skall användas:

- narrow (modul/ratio 2, default)
- regular (modul/ratio 2,5)
- bold (modul/ratio 3)

”enHight”, som anger streckkods höjd

- low (FontSize * 0,1333mm, default)
- medium (FontSize * 0,1833mm)
- high (FontSize * 0,2367mm)

ex.
(`stringEyeDocBarcodeConverter|Interleaved2of5;narrow;low|xsEyeDocBarcodeRule`)

4.6.3.5 *stringDateConverter*

(EB, 2014-08-11, Forms v1.1.0.331)

Konverterar mellan datum och text. Denna tillför stöd för att kunna använda ”d”-funktionen för att lägga till eller ta bort dagar från ett datum. Skriver man ”d1” eller ”d+1” kommer dagens datum användas som grund och sedan läggs en dag till (alltså imorgon), ”d-2” kommer ta bort två dagar (förrgår), bara ”d” kommer ge dagens datum (idag).

Validering av att värdet är ett korrekt datum kommer också göras mellan XML och fält samt tvärt om, är det felaktigt kommer värdet ersättas av en tom sträng.

Parametern användas för att visa ett annat datum i fältet än vad som står i XML-filen. Detta görs genom att t.ex. d20 kommer lägga till 20 dagar på datumet som visas i fältet. Datumet som visas i fältet kommer då vara dagar fram i tiden och skriver man in ett datum i fältet kommer det att läggas till 20 dagar på det datum man skrev in när man lämnar fältet.

4.6.3.6 *stringDoubleConverter*

(EB, 2014-08-11, Forms v1.1.0.331)

Konverterar mellan ett flyttal och text. Validerar att värdet är ett korrekt flyttal och ersätter annars med en tom sträng.

4.6.3.7 *stringEncodedConverter*

(EB, 2014-08-11, Forms v1.1.0.331)

Konverterar värdet rakt av som en text sträng utan någon speciell hantering.

4.6.3.8 *stringEnumBooleanConverter*

(EB, 2014-08-11, Forms v1.1.0.331)

Konverterar värdet mellan en boolean och en specifik textsträng som angetts i Converterns parameter. Denna används för att spara ett specifikt värde i XML-filen om man kryssar i en CheckBox.

Anger man texten ”ja” som parameter och kryssar i CheckBox:en sparas värdet ”ja”. Om det står ”ja” i XML-filen kommer CheckBox:en kryssas i (genom att värdet True returneras). (`stringEnumBooleanConverter|ja`)

Detta gör att man kan använda sig av envalsfunktionalitet på CheckBox:ar. Om man har varsin CheckBox med parametrarna ”ja” respektive ”nej” kopplade till samma värde i XML-filen kommer bara en av dem kunna vara ikryssad i taget. Är värdet något annat än ”ja” eller ”nej” kommer ingen av de två i exemplet vara ikryssade.

Kryssar man ur en CheckBox kommer värdet som sparas bli en tom sträng, vilket blir null i Innehållstjänsten.

Notera att Convertern är skiftlägeskänslig så en kryssruta med parametern ”ja” kommer inte vara ikryssad om värdet är ”Ja”.

4.6.3.9 *stringInt32Converter*

(EB, 2014-08-11, Forms v1.1.0.331)

Konverterar mellan ett heltal och text. Validerar att värdet är ett korrekt heltal och ersätter annars med en tom sträng.

4.6.3.10 *stringRegExpConverter*

(EB, 2014-08-11, Forms v1.1.0.331)

Det finns idag ingen skillnad mellan denna och den vanliga stringEncodedConverter, men möjliggör användandet av xsRegExpRule som också använder angiven parameter.
Referens: Se kapitel 4.6.5.12 XsRegExpRule.

4.6.3.11 *stringTimeConverter*

(EB, 2014-08-11, Forms v1.1.0.331)

Regelverk för inmatning av tid i Forms. Givet inmatad timeString ska det komma ut en tid. Är tiden inte komplett ska det paddas med aktuell tid, sekunder sätts till 00 (hh:mm:00). Klockan ska gå runt, runt, dvs 23:00:00 + 4h -> 03:00:00. 24 timmars klocka avses.

Parameter kan användas på samma sätt som för stringDateConverter, men med ”t” istället för ”d” som tecken.

4.6.3.12 *stringYearConverter*

(EB, 2014-08-12, Forms v1.1.0.331)

Det finns idag ingen skillnad mellan denna och den vanliga stringEncodedConverter, men möjliggör användandet av xsYearRule. **Referens:** Se kapitel 4.6.5.3 XsYearRule.

4.6.4 ConverterParameter

(EB, 2014-08-12, Forms v1.1.0.331)

Tilläggsparametrar tillhörande sin föregående Converter. Tillämpningen varierar så för användning se respektive Converters kapitel under: **Referens:** Se kapitel 4.6.3 Converter.

4.6.5 Rule

(EB, 2014-08-13, Forms v1.1.0.331)

Rule är det som validerar att det värde som står i fältet är av ett korrekt format och/eller datatyp. Till skillnad från Converter så validerar Rule enbart värdet och fält med felaktiga värden markeras med en röd ram.

Rule skyddar också tillsammans med Converter (i förekommande fall) XML-filen från att användaren skriver in felaktiga värden som inte uppfyller definitionen av värdets format eller datatyp i XML-schemat. I dagsläget går det inte att spara blanketten om ett felaktigt värde skickas till Innehållstjänsten, t.ex. bokstäver i ett hel- eller flyttalsvärde.

Rule valideringen sker innan bindingen sparar värdet till XML-filen och om valideringen säger att värdet är felaktigt kommer värdet aldrig sparas till XML-filen. När ett värde sparas till XML-filen kommer bindingen mellan XML och fält automatiskt uppdatera alla fält med dess värden från XML-filen.

Så genom detta uppstår här en skillnad från Converterns validering som tömmer värdet om det skulle vara felaktigt. Rule kommer istället om värdet är felaktigt att markera fältet rött utan att ta bort värdet, men nästa gång XML-filen uppdateras (av en funktion eller om användaren skriver något i ett annat fält) så kommer värdet återställas till det som står i XML-filen, alltså det tidigare värdet.

Valideringen sker enbart när värdet ska sparas till XML-filen, alltså när man lämnar fältet eller när man gör ett val beroende på vilken typ av fält det gäller. Om XML-filen isället innehåller ett felaktigt värde kommer Rule, till skill inte att validera eller markera det om inte fältet uppdateras av användaren.

Markeringen att fältet innehåller ett fel görs med hjälp av en Style som kontrollerar fätets Validation.HasError property och finns definierat enbart för TextBox och ComboBox.

Följande är de regler är som finns tillgängliga och kan användas som Rule:

Regel	Converter
XsDateRule	StringDateConverter
XsSqlDateRule	StringDateConverter
XsYearRule	StringYearConverter
XsDoubleRule	StringDoubleConverter
XsInt32Rule	StringInt32Converter
XsCode128Rule	StringCode128Converter, stringEyeDocBarcodeConverter, StringEncodedConverter
XsEyeDocBarCodeRule	stringEyeDocBarcodeConverter, StringEncodedConverter
XsPrescribeCodeRule	StringPrescribeCodeConverter, StringEncodedConverter
XsWorkplaceCodeRule	StringWorkplaceCodeConverter, StringEncodedConverter
XsPrescribeAndWorkplaceCodeRule	StringPrescribeAndWorkplaceCodeConverter, StringEncodedConverter
XsTimeRule	StringTimeConverter
XsRegExpRule	StringRegExpConverter

Notera att varje regel enbart fungerar tillsammans med rätt Converter och kan inte användas i någon annan kombination än vad tabellen visar. Detta betyder dessutom att vissa regler enbart kan användas till vissas fält enligt tabellen: **Referens:** Se kapitel 4.6.3 Converter.

Förutom dessa standardregler finns också utökade valideringsregler som placeras i ExtendedRule, **Referens**: Se kapitel 4.19 Validering.

4.6.5.1*XsDateRule*

(EB, 2014-08-14, Forms v1.1.0.331)

Validerar att värdet i fältet är ett korrekt datum eller texten ”d” eventuellt följt av ett positivt eller negativt heltal (t.ex. ”d3”, ”d+5”, ”d-7”). Godkända datum är ”0000-01-01” till ”9999-12-32”.

4.6.5.2 *XsSqlDateRule*

(EB, 2014-08-14, Forms v1.1.0.332)

Validerar att värdet i fältet är ett korrekt datum eller texten ”d” eventuellt följt av ett positivt eller negativt heltal (t.ex. ”d3”, ”d+5”, ”d-7”). Godkända datum är ”1753-01-01” till ”9999-12-32” för att matcha godkända datum i SQL Server. Dock finns också skydd för att användaren skulle skriva ”d-100000” vilket från ”2014-08-01” är ”1740-10-29”.

XsSqlDateRule bör användas istället för xsDateRule om värdet sparas in en nod i XML-filen som enligt schemat är av en datumtyp då det betyder att Innehållstjänsten sparar värdet som ett datum och inte en textsträng i databasen.

4.6.5.3 *XsYearRule*

(EB, 2014-08-14, Forms v1.1.0.331)

Validerar att värdet i fältet är ett korrekt år. Godkända år är ”1753” till ”9999”. Notera att inget stöd finns för ”d” funktionen.

4.6.5.4 *XsDoubleRule*

(EB, 2014-08-14, Forms v1.1.0.331)

Validerar att värdet i fältet är ett korrekt flyttal.

4.6.5.5 *XsInt32Rule*

(EB, 2014-08-14, Forms v1.1.0.331)

Validerar att värdet i fältet är ett korrekt heltal.

4.6.5.6 *XsCode128Rule*

(EB, 2014-08-14, Forms v1.1.0.331)

Validerar att värdet är ett numeriskt tal på 12, 14, 16, 18 eller 20 siffror.

4.6.5.7 *XsEyeDocBarCodeRule*

(EB, 2014-08-14, Forms v1.1.0.331)

Validerar att värdet följer den regular expression som gäller för den streckkod som är angiven i fältets ConverterParameter. **Referens:** Se kapitel 4.6.3.4 stringEyeDocBarcodeConverter.

4.6.5.8 *XsPrescribeCodeRule*

(EB, 2014-08-14, Forms v1.1.0.331)

Validerar att värdet är ett numeriskt tal på 7 siffror.

4.6.5.9 *XsWorkplaceCodeRule*

(EB, 2014-08-14, Forms v1.1.0.331)

Validerar att värdet är ett numeriskt tal på 3, 5, 7, 11 eller 13 siffror.

4.6.5.10 *XsPrescribeAndWorkplaceCodeRule*

(EB, 2014-08-14, Forms v1.1.0.331)

Validerar att värdet är ett numeriskt tal på 10, 12, 14, 16, 18 eller 20 siffror.

4.6.5.11 *XsTimeRule*

(EB, 2014-08-14, Forms v1.1.0.331)

Validerar att värdet i fältet är en korrekt tid eller texten ”t” eventuellt följt av ett positivt eller negativt heltal (t.ex. ”t3”, ”t+5”, ”t-7”).

4.6.5.12 *XsRegExpRule*

(EB, 2014-08-14, Forms v1.1.0.331)

Validerar att värdet följer den regular expression som finns angiven i fältets ConverterParameter.

4.6.6 ExtendedRule

(EB, 2015-08-06, Forms v1.1.0.344)

ExtendedRule är utökade valideringsregler. För att kunna använda sig dessa måste validering aktiveras i Formsettings. **Referens:** Se kapitel 4.5.3 Validation.

Varje ExtendedRule består av tre parametrar som separeras med semikolon (;). Den tredje parametern är underparametrar till valideringsregeln, dessa separeras med kolon (:). Notera att samtliga parametrar (semikolon) måste finnas med även om de inte används.

```
<Tag>
  ElementName|XPath|Converter|ConverterParameter|Rule
  |ValidationLevel;ExtendedRule;Param:Param:Param
</Tag>
```

Parameter	Beskrivning
ValidationLevel	Vilken valideringsnivå som regeln ska använda. **Referens:** Se kapitel 4.19.1 Valideringsnivå.
ExtendedRule	Namnet på vilken valideringsregel som ska användas. **Referens:** Se kapitel 4.19.6 Valideringsfunktioner.
Param	Underparametrar som ska användas. Dessa är unika per varje valideringsregel. **Referens:** Se kapitel 4.19.6 Valideringsfunktioner.

För att använda ytterligare valideringsregler separeras dessa med vertikalstreck (|).

Underparametrar kan enbart utelämnas som de inte har någon efter sig som ska användas eller är obligatorisk. Annars måste samtliga kolon (:) finnas med. Om en parameter får utelämnas, vara tom eller ha ett felaktigt värde beror på respektive regel.

Referens: Se kapitel 4.19.6 Valideringsfunktioner för information om hur respektive valideringsregel ska definieras.

4.6.7 Encoding och decoding

(EB, 2014-08-13, Forms v1.1.0.331)

Notera att detta kapitel enbart gäller för tecken i Tag-attributet, för övrigt i XAML används normal XML-encoding, med t.ex. ”<” för ”<”. Men eftersom ”;” inte kan användas i ElementTag, då den används som parameterseparator, finns en egen hantering för detta enligt nedan.

Vissa tecken som kan användas i en parameter i en regel måste skrivas om för att kunna användas. Som exempel kan tecknet ”|” behöva användas som pattern i en RegExp, liknande exempel finns för StringFormat.

Följande tecken är några av de som är reserverade:

- | = Vertikalstreck (pipe)
- ; = Semikolon
- : = Kolon
- & = Och
- < = Mindre än
- > = Större än
- ” = Citat
- ’ = Apostrof

De tre första används som separatorer av ElementTag och övriga är reserverade XML-tecken.

För att hantera dessa tecken används nedan tre funktioner, den gamla hårdkodade funktionen, en generisk decimal-baserad funktion och en hex-baserad funktion.

Format	Prio	Beskrivning
\u000000u	Körs först	Börjar med \u, sedan ett decimalvärde från ”0” och uppåt som motsvarar ett unicode (UTF-32) tecken, avslutas med u\
\uoooo		Hårdkodade tecken enligt tabellen nedan.
\uoooo	Körs sist	Börjar med \u, senan ett 4 siffrigt hex-värde från ”0000” till ”FFFF” som motsavarar ett unicode (UTF-16) tecken, behöver inte avslutas utan måste alltid vara ha samma längd.

4.6.7.1 *Decimal-format*

(EB, 2014-08-13, Forms v1.1.0.331)

Anledningen till att dessa måste avslutas med ”u\” är för att det ska vara möjligt att använda mer än 4 siffror. Detta medför att samtliga tecken kan användas då det inte finns någon begränsning på antal siffror. I praktiken betyder detta upp till ”1114111” (hex 10FFFF).

Referens: För gränsvärden och hur konverteringen sker se System.Char.ConvertFromUtf32 Method.
Referens: För lista av hex-värden för UTF32 se Complete Character List for UTF-32.

Notera dock att enbart decimalvärden kan användas, ej hex vilket oftast används i listor över tecken. Konvertering mellan hex och decimal kan göras med t.ex. kalkylatorn i Windows genom att välja Visa > Programmerare och sedan byta mellan knapparna Hex och Dec med ett värde inskrivet i räknaren.

Nedan följer några exempel på tecken som kan användas:

Tecken	Ersättning	Hex	Decimal
| (vertikalis)	\u124u\	7c	124
; (semikolon)	\u59u\	3b	59
: (kolon)	\u58u\	3a	58
& (och)	\u38u\	26	38
> (större än)	\u62u\	3e	62
< (mindre än)	\u60u\	3c	60
” (citat)	\u34u\	22	34
’ (apostrof)	\u39u\	27	39

Exempel: en parameter som ska vara ”`True|False`” måste skrivas ”`True\u124u\False`”.

4.6.7.2 *Hex-format*

(EB, 2014-08-13, Forms v1.1.0.331)

Även om decimal-funktionen klarar samtliga* UTF-32 tecken finns denna funktion som representerar det som den gamla funktionen borde gjort från början. Funktionens ersättningsvärden ska skrivas i hex-format istället för decimal för att få med fler tecken inom de tillgängliga fyra värde-tecknen som går att använda.

Anledningen till att det finns en värde-teckens begränsning är för att ersättningen inte innehåller något sluttecken och därför måste ha en fast längd för att inte råka ta med ett tecken efter som inte ska vara en ersättning av (t.ex. b>a = b\u0001a).

Att det skrivs i Hex och att fyra värdetecken används ger att värdena ”0000” till ”FFFF” går att använda vilket är 65536 tecken (= 2^16 = 16 bit = 2 byte), alltså UTF-16.

Referens: För lista av hex-värden för UTF32 se Complete Character List for UTF-32. Notera att enbart UTF-16 tecken enligt ovan kan användas här.

Följande är några exempel på ersättningar för de reserverade tecken.

Tecken	Ersättning	Hex	Decimal
| (vertikalis)	\u007c	7c	124
; (semikolon)	\u003b	3b	59
: (kolon)	\u003a	3a	58
& (och)	\u0026	26	38
> (större än)	\u003e	3e	62
< (mindre än)	\u003c	3c	60
” (citat)	\u0022	22	34
’ (apostrof)	\u0027	27	39

Exempel: en parameter som ska vara ”`True|False`” måste skrivas ”`True\u007cFalse`”.

Denna funktion kommer alltid köras sist för att inte störa den gamla funktionen, vilket betyder att följande tecken inte kan användas.

Tecken	Ersättning	Hex	Decimal
Ĥ	\u0124	124	292
Y	\u0059	59	89
X	\u0058	58	88
8	\u0038	38	56

4.6.7.3 *Gammalt format (decimal)*

(EB, 2014-08-13, Forms v1.1.0.331)

De gamla ersättningsvärdena bevaras eftersom det används i gamla blanketter. Dessa värden är skrivna i decimal-format när det egentligen skulle ha varit hex, eftersom 4-tecken i hex är 16-bitars tecken (2 byte) = 0-65535. T.ex. ”|” har hex-värdet 7C, men decimal 124. På fyra tecken hade det då gått att använda upp till 9999 decimal = 270F hex = ✏ (penna). Även om det troligen hade varit tillräckligt finns nu i vilket fall en mer ”generisk” lösning som klarar samtliga tecken och inte bara de fyra förbestämda tecken som inte inkluderar alla reserverade XML-tecken.

Följande tecken hanteras fortfarande av den gamla funktionen för bakåtkompabilitet:

Tecken	Ersättning
| (Vertikalstreck/pipe)	\u0124
; (Semikolon)	\u0059
: (Kolon)	\u0058
& (Och)	\u0038

Exempel: en parameter som ska vara ”`True|False`” måste skrivas ”`True\u0124False`”.

4.7 Signering

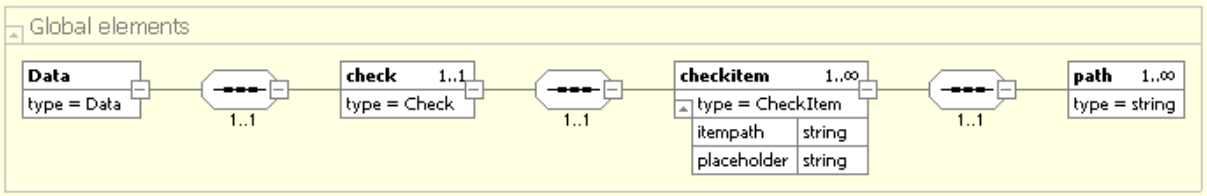
4.7.1Mjuk signering/historik

För att se vem som senast fyllt i ett avsnitt används mjuk signering.

4.7.1.1 *Design*

4.7.1.2 *Beskrivning XAML*

Nedan beskrivs tilläggen som ska göras i XAML filen. Placera först en `XmlDataProvider`, med namn `checkData`, först i filen(`StackPanel.Resources`), med följande schema:



Figur: Schema check i XAML

Namn	Beskrivning
itemPath	Sökvägen till usercheck i XML
placeholder	Namnet på den stackpanel, där informationen om användare ska placeras.
Path	Sökvägen till de element som ska ingå. Sökvägen kan vara partiell.

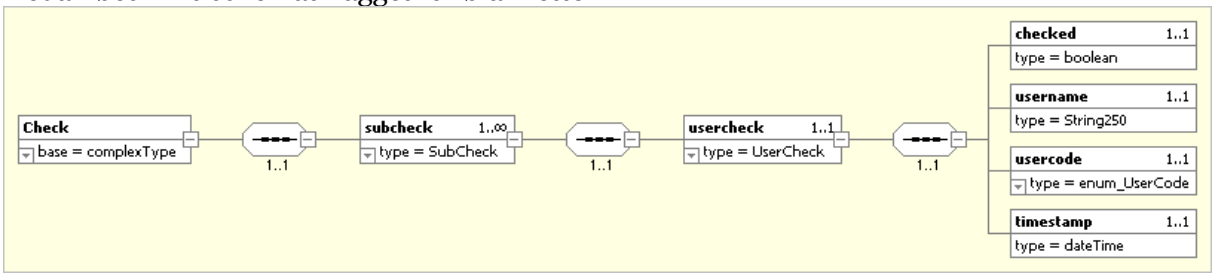
Tabell: Förklaring attribut och noder.

Exempel:

```
<XmlDataProvider x:Key="checkData" XPath="/Data">
  <x:XData>
    <Data xmlns="">
      <check>
        <checkitem itempath="check/subcheck[1]/usercheck" placeholder="placeholderCheck1">
          <path>cgoperationlist[1]/cgactivitylist[1]/cgbooleanlist[1]/booleanlist</path>
          <path>cgoperationlist[1]/cgactivitylist[1]/cgenumlist[1]/enumlist/enumlistvalue/value</path>
        </checkitem>
      </check>
    </Data>
  </x:XData>
</XmlDataProvider>
```

4.7.1.3 *Beskrivning XML*

Nedan beskrivs schematillägget för blanketten



Figur: Schema check i XML

Namn	Beskrivning
checked	Sätts till sant vid ändring
username	Sätts till användarens namn, vid ändring.
usercode	Sätts till användarens yrkeskod, vid ändring.
timestamp	Sätts automatisktvid ändring.

Tabell: Förklaring noder.
Exempel:

```
<check>
  <subcheck>
    <!--  Aktivitet (1) -->
    <usercheck>
      <checked>false</checked>
      <username/>
      <usercode/>
      <timestamp/>
    </usercheck>
  </subcheck>
</check>
```

4.7.2 SignLock

```
<Border x:Name="elem_border_signature_document" Width="374" Background="#ffffffff"
BorderBrush="#ff000000" BorderThickness="1.5" CornerRadius="4" Visibility="Visible">

   <StackPanel x:Name="DocumentSignedStackPanel" Margin="2,4,6,2" Visibility="Collapsed">

     <WrapPanel>
       <TextBlock Text="Signerad/Låst av:" FontSize="10" Width="100" Style="{DynamicResource
tblStyle}" VerticalAlignment="Center"/>
       <TextBlock x:Name="DocumentSignedByName" Style="{DynamicResource
tblStyle}"></TextBlock>
     </WrapPanel>

     <WrapPanel>
       <TextBlock Text="Titelkod:" FontSize="10" Width="100" Style="{DynamicResource
tblStyle}" VerticalAlignment="Center"/>
       <TextBlock x:Name="DocumentSignedByTitleCode" Style="{DynamicResource
tblStyle}"></TextBlock>
     </WrapPanel>

     <WrapPanel>
       <TextBlock Text="Tidstämpel:" FontSize="10" Width="100" Style="{DynamicResource
tblStyle}" VerticalAlignment="Center"/>
       <TextBlock x:Name="DocumentSignedTimeStamp" Style="{DynamicResource
tblStyle}"></TextBlock>
      </WrapPanel>

   </StackPanel>
   <Border.RenderTransform><TranslateTransform X="356" Y="916"/></Border.RenderTransform>


</Border>
```

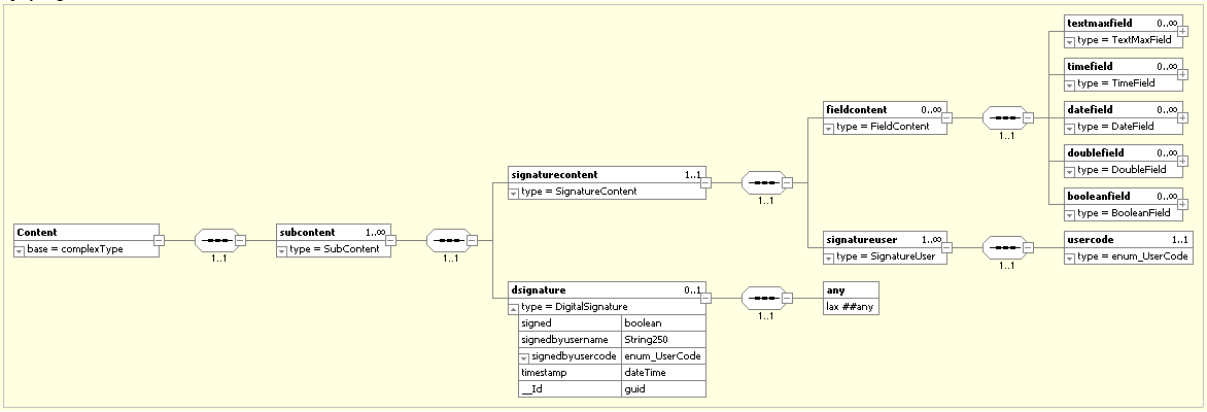
SignLock använder ett nytt sätt att Hel-signera i formsblanketten (eId webbtjänsten). För att funktionen ska vara tillgänligt ska det finnas en SignLock parametern i Formsettings och att i blanketten finns färdigreserverat fält för ändamålet. De fyra element som behövs för att funktionen fungera, finns beskrivet i formsettingen.

Dessutom krävs att version 2 av valideringsfunktionen är aktiv (se dess kapitel för mer info).

4.7.3 Delsignering

Delsignering används framförallt i standardvårdplaner.

4.7.3.1*Schema*



Figur: Schema Content

4.7.3.2 *Delar av schemat i kod*

```
<complexType name="SignatureContent">
  <sequence>
    <element name="fieldcontent" type="tns:FieldContent" minOccurs="0" maxOccurs="unbounded"/>
    <element name="signatureuser" type="tns:SignatureUser" minOccurs="1" maxOccurs="unbounded"/>
  </sequence>
  <attribute name="Id" type="tns:guid"/>
  <attribute name="__Id" type="tns:guid"/>
</complexType>
<complexType name="DigitalSignature">
  <sequence>
    <any namespace="##any" processContents="lax" minOccurs="0"/>
  </sequence>
  <attribute name="signed" type="boolean"/>
  <attribute name="signedbyusername" type="tns:String250"/>
  <attribute name="signedbyusercode" type="tns:enum_UserCode"/>
  <attribute name="timestamp" type="dateTime"/>
  <attribute name="__Id" type="tns:guid"/>
</complexType>
<complexType name="SubContent">
  <sequence>
    <element name="signaturecontent" type="tns:SignatureContent" minOccurs="1" maxOccurs="1"/>
    <element name="dsignature" type="tns:DigitalSignature" minOccurs="0" maxOccurs="1"/>
  </sequence>
  <attribute name="__Id" type="tns:guid"/>
</complexType>
<complexType name="Content">
  <sequence>
    <element name="subcontent" type="tns:SubContent" minOccurs="1" maxOccurs="unbounded"/>
  </sequence>
  <attribute name="__Id" type="tns:guid"/>
</complexType>
```

4.7.3.3 *XML signering*

Blankett XML:et, är delvis ifyllt och det är viktigt att `Id` attributet till `signaturecontent` är unikt. Attributet `signed` till `dsignature` ska vara `0` alternativt `false` om den gamla kuvertsigneringen används.

När signeringen är signerad sätts attributet `signed` i `dsignature` till `1` alternativt `true` om den gamla kuvertsigneringen används.

Anledningen till att `1` och `0` ska användas är att värdet `signed` är en Boolean och Innehållstjänsten lämnar ifrån sig 0 och 1.

4.7.3.4 *Exempel på kod i XML för blankett*

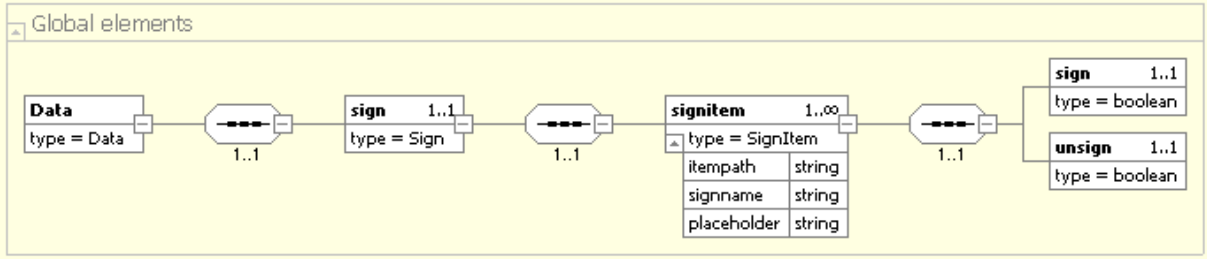
Notera att värdet på attributet Id, ska vara ett unikt guid, skrivet med gemener.

```
<content>
  <!-- (1) -->
  <subcontent>
    <!-- Hud/Vävnad observation -->
    <signaturecontent Id="{36b81d90-3947-11dd-ae16-0800200c9a66}">
      <!-- 1 -->
      <fieldcontent>
        <datefield>
          <value/>
        </datefield>
        <doublefield>
          <value/>
        </doublefield>
        <booleanfield>
          <value/>
        </booleanfield>
      </fieldcontent>
      <signatureuser>
        <usercode>*</usercode>
      </signatureuser>
    </signaturecontent>
    <dsignature signed="false" signedbyusercode="*" signedbyusername="" timestamp="2007-01-
01T00:00:00Z">
      <value/>
    </dsignature>
  </subcontent>
</content>
```

4.7.3.5 *XAML*

Införa en XML resurs i XAML-filen, som beskriver kopplingen mellan XML och blanketten.

4.7.3.6 *Schema*



Figur: Schema sign resurs

4.7.3.7*Schema som kod*

```
<?xml version="1.0" encoding="utf-8"?>
<schema targetNamespace="http://gainit.se/Forms/sign/xaml"
  attributeFormDefault="unqualified"
  elementFormDefault="qualified"
  xmlns:tns="http://gainit.se/Forms/sign/xaml"
  xmlns="http://www.w3.org/2001/XMLSchema">
  <complexType name="SignItem">
    <sequence>
      <element name="sign" type="boolean">
        <annotation>
          <documentation>If we want to sign the item.</documentation>
        </annotation>
      </element>
      <element name="unsign" type="boolean">
        <annotation>
          <documentation>If we want to unsign the item.</documentation>
        </annotation>
      </element>
```

```
    </sequence>
  </complexType>
  <complexType name="Sign">
    <sequence>
      <element name="signitem" type="tns:SignItem"/>
    </sequence>
    <attribute name="itempath" type="string">
      <annotation>
        <documentation>The XPath to the sign subcontent, to place sign info.</documentation>
      </annotation>
    </attribute>
    <attribute name="signname" type="string">
      <annotation>
        <documentation>The name of the checkbox, used when we want to sign the content.</documentation>
      </annotation>
    </attribute>
    <attribute name="placeholder" type="string">
      <annotation>
        <documentation>The name of the stackpanel, to place XAML-code.</documentation>
      </annotation>
    </attribute>
  </complexType>
  <complexType name="Data">
    <sequence>
      <element name="sign" type="tns:Sign"/>
    </sequence>
  </complexType>
  <element name="Data" type="tns:Data"/>
</schema>
```

4.7.3.8 *Uppbyggnad av XML i XAML*

```
<XmlDataProvider x:Key="signData" XPath="/Data">
  <x:XData>
    <Data xmlns="">
      <sign>
        <signitem itempath="content/subcontent[1]" signname="elemSign1" placeholder="placeholderSign1"
deviationname="elemDeviation1" signlevel="Session" signtype="Single">
          <sign>false</sign>
          <unsign>false</unsign>
        </signitem>
      </sign>
    </Data>
  </x:XData>
</XmlDataProvider>
```

`itempath` = Adress till den plats i XML:et där signeringen skall sparas. Måste innehålla d, samt om

`signname` = Namn på den kryssruta som skall aktivera signeringen.

`deviationname` = Namn på den kryssruta som skall aktivera avvikelsesignering.

`placeholder` = Namn på den StackPanel som skall fyllas med info om signeringen för användaren.

`signlevel` = Vilken nivå av signering som ska användas: `Session` = signering på sessionsuppgifter på den inloggade användaren, `Advanced` = signering med användarnamn och lösenord, `Qualified` = signering med certifikat, t.ex. SITHS-kort. Anges inte `signlevel` är `Advanced` default.

`signtype` = Vilken typ av signering som ska användas: `Batch` = signering görs när användaren klickar på signeringsknappen i menyn, `Single` = signering görs direkt när kryssrutan kryssas i. Anges inte `signtype` är `Batch` default.

Om avvikelsesignering skall användas så måste kryssrutorna för både sign och avvikelse ha xPath. I dagsläget fungerar enbart `signlevel`/`signtype` i kombinationerna `Advanced`/`Batch` och `Session`/`Single`.

4.7.3.9 *Struktur i XAML*

```
<WrapPanel Margin="0,0,0,2">
  <WrapPanel IsEnabled="{Binding ElementName=elemsign1, Path=IsEnabled}">
    <TextBox Margin="0,0,10,0" Style="{DynamicResource tbStyle}" Name="elem543" Width="100"
MaxWidth="100" Validation.ErrorTemplate="{DynamicResource validationTemplate}">

<TextBox.Tag>|content/subcontent[1]/signaturecontent/fieldcontent/datefield/value|stringDateConverter||
xsDateRule|</TextBox.Tag>
    </TextBox>
    <TextBox Margin="0,0,10,0" Style="{DynamicResource tbStyle}" Name="elem544" Width="100"
MaxWidth="100" Validation.ErrorTemplate="{DynamicResource validationTemplate}">

<TextBox.Tag>|content/subcontent[1]/signaturecontent/fieldcontent/doublefield/value|stringDoubleConvert
er||xsDoubleRule</TextBox.Tag>
    </TextBox>
  </WrapPanel>
  <CheckBox Margin="5,0,43,0" Style="{DynamicResource cbStyle}" Name="elemsign1" Width="12"/>
  <StackPanel x:Name="placeholdersign1"/>
</WrapPanel>
```

4.7.3.10 *Låsa fält i XAML*

Från version 268 så låses inte längre kryssrutorna för signering för att det ska vara möjligt att göra avsigneringar. Därför kan inte signerat innehåll låsas på samma sätt längre utan följande måste göras:

Koppla en kryssruta mot värdet om signeringen är signerad eller inte med en `stringEnumBooleanConverter` till 0 enligt exemplet:

```
<CheckBox x:Name="Sign2" Style="{DynamicResource cbStyle}" Visibility="Hidden">
<CheckBox.Tag>|content/subcontent[1]/dsignature/@signed|stringEnumBooleanConverter|0</CheckBox.Tag>

</CheckBox>
```

Koppla igen `IsEnabled` på det som ska låsas men nu mot kryssrutan i föregående steg med `IsChecked`:

```
IsEnabled="{Binding ElementName=Sign2, Path=IsChecked}"
```

4.7.4 Helsignering

Helsignering är signering som signerar hela blanketten. Vid signering kommer blankettens status att ändras till en låst status för signering och inget i blanketten kan därefter ändras. Detta kan man sedan koppla mot andra funktioner som t.ex. skickar blanketten eller dess data vidare till en extern part och då med krav att signering görs innan den skickas.

Helsignering är en ögonblicksbild av blanketten, som signeras. I ett första försök hanterar vi bara helsignering i ett dokument utan delsignering. (lösa FK 7263)
Frågor att fundera på:

- Ska vi tillåta ändring av blanketten efter helsignering?
- Hur vi ska hantera helsignering när det finns delsignering?
- Kan man helsignera när bara några delsigneringar är signerade?

4.7.4.1 *Exempel på kod i XML för blankett*

```
<dsignature signed="false" signedbyusercode="*" signedbyusername="" timestamp="2007-01-01T00:00:00Z">
```

4.7.5 Definition av XML i XAML

Vi lägger en definition i `formSettings`(i XAML-filen) som talar om att dokumentet kan helsigneras. Vi behöver XPath:en till attributet. Ritningen av signeringen lägger i direkt i XAML-koden, se nedan.

```
<signdocument signedpath="dsignature/attribute::signed"/>
```

4.7.5.1 *Struktur i XAML*

```
<Border x:Name="elem_border_signature_document" Width="374" Background="#ffffffff"
BorderBrush="#ff000000" BorderThickness="1.5" CornerRadius="4" Visibility="Visible">
  <Border.Tag>|dsignature/attribute::signed|stringBooleanVisibilityConverter|</Border.Tag>
  <StackPanel Margin="2,2,6,2">
    <WrapPanel>
      <TextBlock Text="Signerad av:" FontSize="10" Width="90" Style="{DynamicResource tblStyle}"
VerticalAlignment="Center"/>
      <TextBlock Text="{Binding XPath=dsignature/attribute::signedbyusername}" Style="{DynamicResource
tblStyle}"/>
    </WrapPanel>
    <WrapPanel>
      <TextBlock Text="Titelkod:" FontSize="10" Width="90" Style="{DynamicResource tblStyle}"
VerticalAlignment="Center"/>
      <TextBlock Text="{Binding XPath=dsignature/attribute::signedbyusercode}" Style="{DynamicResource
tblStyle}"/>
    </WrapPanel>
    <WrapPanel>
      <TextBlock Text="Tidstämpel:" FontSize="10" Width="90" Style="{DynamicResource tblStyle}"
VerticalAlignment="Center"/>
      <TextBlock Text="{Binding XPath=dsignature/attribute::timestamp}" Style="{DynamicResource
tblStyle}"/>
    </WrapPanel>
  </StackPanel>
  <Border.RenderTransform>
    <TranslateTransform X="356" Y="916"/>
  </Border.RenderTransform>
</Border>
```

4.8 Lock

(EB, 2015-05-05, Forms v1.1.0.339)

Funktionen Lock består i grunden av fyra åtgärder som kan kombineras enligt följande tabell vilket också beskriver de möjlig lägen som funktionen kan vara i:

Parameter	Funktion	Beskrivning
None	-	Ingen åtgärd.
Lock	Lås	Ärendet låses med status 30.
Close	Arkivera	Ärendet arkiveras med bit status 8192.
LockAndClose	Lås och arkivera	Både låsning och arkivering enligt föregående.
Unlock	Lås upp	Låser upp ärendet genom att plocka bort status 30, detta fungerar dock inte idag.

Open	Öppna	Tar bort arkiveringen från ärendet genom att plocka bort bit status 8192.
UnlockAndOpen	Lås upp och öppna	Både

Det går inte att använda funktionerna för att låsa och arkivera i samma blankett så att båda kan utföras separat från varandra. Det finns helt enkelt enbart en menyknapp och en metod för att utföra båda. Så antingen får man använda den ena, den andra eller både på en gång. Detta är mest för att underlätta för användaren och undvika för många knappar då båda funktionerna egentligen betyder samma sak, att ärendet inte längre är aktivt. Där lås avslutar, men utan signering och där arkivera bara gömmer ärendet från listningar. Vilket får användas utifrån den kombination som passar för blanketten i fråga.

Utifrån vilken status blanketten har:
- Om den är låst/signerad
- Om den är arkiverad

Och vilka åtgärder som är aktiverade i blanketten:
- Lås
- Arkivera
- Lås upp
- Öppna

Så plockas ett aktivt läge fram enligt tabellen ovan.

Utifrån aktivt läge så sätts menyknappens ikon och tooltip för att matcha detta. Samma metod används sedan för att avgöra vad som ska hända när användaren faktiskt klickar på knappen för att avgöra vilka texter som ska användas i bekräftelsedialogen och vilken/vilka åtgärder som ska utföras.

Följande gäller för funktionerna:
- Åtgärderna utförs genom att parametrar sätts utifrån aktivt läge.
- Dessa parametrar skickas sedan i ett spara-XML anrop till Kuvertet.
- Ett makulerat ärende kan varken låsas eller arkiveras.
- Ärendet måste vara sparat för att låsas eller arkiveras. Detta delvis för att valideringar m.m. ska köras, även om funktionen i praktikens skulle kunna spara automatiskt, vilket kan vara en förbättringspunkt.

Låsningen sparar ärendet med status och sätter den till låst/signerad (30).
- Blanketten blir låst och kan inte ändras.
- En signerad blankett kan inte låsas eller låsas upp, eftersom signeringen är sin egen och en avsignering skulle behövas för att låsa upp vilket måste hanteras av signeringsfunktionen (SignLock).
- En blankett som redan är låst, på ett eller annat sätt, kan inte låsas.

- Idag kan inte heller en blankett låsas upp, eftersom Kuvertet inte godkänner att ett ärende med status 30 lämnar den statusen. För detta skulle istället en ny ingång behövas specifikt för ändamålet.
- Låsning är aktivt för alla blanketter som inte har SignLock, men kan aktiveras även på dessa.

Arkiveringen lägger till bitstatus Arkiverad (8192) till ärendet.

- Detta betyder att det visas som arkiverat som listningar och göms från listningar som visar aktiva ärenden.
- Blanketten går fortfarande skriva i och spara om den inte är låst på annat sätt.
- Arkivering kan göras på signerade och skickade ärenden.
- Arkivering är inte aktiverat som standard utan måste aktiveras i blankettinställningarna.

Referens: Se kapitel 4.5.5 Lock för blankettinställningar.

4.9 Makulera

(EB, 2014-08-07, Forms v1.1.0.330)

Makuleringsfunktionen makulerar hela ärendet. Detta görs genom ett anrop till Kuvertet som sätter bitstatusen för makulerad (bit 4, decimal 8, binary 1000).

4.9.1Makuleringsmarkering

(EB, 2014-08-07, Forms v1.1.0.330)

En makuleringsmarkering kommer att visas på blanketten om den har bitstatusen makulerad. Detta fungerar enbart för blanketter som har FixedPage.

4.10 Träd

(EB, 2011-02-21, Forms v1.1.0.178)

Trädet kan innehålla tre olika typer av noder som har olika utseende och funktion.
`RootHeaderItem` är tänkt motsvara en blankett och kan visa två värden vilket bör vara blankettens benämning och artikelnummer.
`HeaderItem` är tänkt att motsvara en rubrik och kopplas normal antingen mot en namngiven rubrik (t.ex. en Glyphs eller TextBlock) eller det första fältet under rubriken. Den kan i sin tur innehålla underelement om det finns underrubriker eller fält under den eller båda delarna.
`Item` är ett slutgiltigt element som inte kan innehålla några under delar. Detta bör anses vara ett fält i blanketten.

Det finns i Forms ingen fysisk begränsning hur de används mer än hur deras grundläggande funktionalitet är vilket betyder att det finns stora friheter för hur de kan användas. Samtidigt är det för användarvänlighetens skull däremot viktigt att trädet byggs upp på ett logiskt sätt, även om det i slutändan är upp till den som beställt blanketten hur den ska se ut och hur trädet ska vara uppbyggt.

I normala fall bör dock `RootHeaderItem` vara det yttersta elementet och endast förekomma EN gång per blankett. `HeaderItem` kan användas både självstängande och med andra `HeaderItem` och `Item` som underelement. Det bör inte heller vara för många nivåer i trädet då det blir svårt för användaren att då får den överblick som det är tänkt att trädet ska ge. Riskerar man att få för många nivåer kan det vara värt att fundera på om verkligen varje fält behöver finnas i trädet eller om det räcker att gruppen finns, då kanske som ett `HeaderItem` för att indikera att det är separat grupp som det pekas på. Samma sak gäller om det blir väldigt många `Item` i en nivå, då det kanske kan vara aktuellt att slå samman flera så att ett `Item` t.ex. adress representerar fälten gatuadress, postnr och postadress. `HeaderItem` kan i vissa fall välja att användas kopplat direkt för ett separat fält om det är viktigt visa det eller om fältet är en del av en rubrik.

Trädet placeras tillsammans med övriga resurser i `StackPanel.Resources` enligt:

```
<XmlDataProvider x:Key="myData" XPath="/Data">
    <x:XData>
        <Data xmlns="">
            <RootHeaderItem Name1="Benämning"  Name2="ArtNr" Sync="txt1">
                <HeaderItem Name="Rubrik 1" Sync="txt2">
                     <Item Name="Fält 1" Sync="txt3"/>
                </HeaderItem>
                <HeaderItem Name="Rubrik 2" Sync="txt2"/>
            </RootHeaderItem>
        </Data>
    </x:XData>
</XmlDataProvider>
```

Forms i EyeDoc 2.0 finns en begränsning att `x:Key="myData"`, för v3.1 (Forms v178) och högre kan även `x:Key="treeData"` användas. `Sync` är det `x:Name` som elementet är kopplat till och går då att koppla till de flesta typer av objekt. Från och med v294 kommer elementet som är kopplat att få fokus när man klickar på noden i trädet, dessutom kommer överkanten av elementet centreras på blanketten oberoende av fönstrets storlek.

I v178 av Forms kan man helt utelämna trädet för att inte visa det för användaren i blanketten (kan eventuellt fungera med tidigare versioner också). Samtidigt kommer då inte heller övriga flikar till vänster att laddas och kommer inte vara åtkomliga för användaren i blanketten (ICD10, versionshistorik).

För information om hur trädet kan används vid signering se kapitlet **Hårdsignering**.

4.10.1 Synkronisering med kryssruta

Av nedanstående funktioner fungerar **Error! Reference source not found.** bäst, det går dock att klicka på den gråa menyn. Dessa fungerar för `Item`, `HeaderItem`, `SignableItem` och `SignableHeaderItem`.

4.10.1.1 *Låsa/låsa upp meny*

För att gråmarkera en meny givet en kryssruta används följande:

```
<HeaderItem Name="Aktivitet Operation 1" Sync="elem70" EnabledSync="CheckBoxOperation1"
EnabledSyncValue="False">
```

4.10.1.2 *Gömma/visa meny*

För att gömma/visa en meny, givet en kryssruta används följande:

```
<HeaderItem Name="Aktivitet Operation 1" Sync="elem70" VisibleSync="CheckBoxOperation1"
VisibleSyncValue="False">
```

4.10.2 Visning av signeringsmarkeringar

Vid användning av delsignering (se signeringsbilaga) så går det att koppla trädet till att visa vilka delar som är signerade och inte. Detta förutsätter att det finns en `SignableHeaderItem` som representerar en del av blanketten och under denna en eller flera `SignableItem`, men endast en för varje delsignering som finns i den representerade delen. Det går däremot förutom detta kombinera fritt med underliggande `HeaderItem` och `Item` vilka inte kommer störa funktionen. Men en varken `SignableHeaderItem` eller `SignableItem` bör någonsin ligga under en `HeaderItem` eftersom detta gör att summering av signeringar inte kommer att fungera eller visas korrekt. Finns en `SignableItem` i trädet ska alltså alla dess föräldrar upp till `RootHeaderItem` vara `SignableHeaderItem`.

```
<RootHeaderItem Name1="Blankettens namn"  Name2="Blankettens nummer" Sync="pat_id">
    ...
    <SignableHeaderItem Name="Undersökning" Sync="Undersokning" ChildrenSigned="None">
        <SignableHeaderItem Name="Sjukdomar" Sync="Sjukdomar" ChildrenSigned="None">
            <Item Name="Cancer" Sync="Cancer"/>
            <SignableItem Name="Signering" Sync="SignCheck" Signed="False"/>
        </SignableHeaderItem>
        ...
        <HeaderItem Name="Behandling" Sync="Behandling">
            <Item Name="Medicin" Sync="Medicin"/>
            <SignableItem Name="Signering" Sync="SignCheck" Signed="False"/>
        </HeaderItem>
    </SignableHeaderItem>
    ...
</RootHeaderItem>
```

Varje `SignableItem` måste sedan vara kopplat med `Sync` till den tillgörande kryssrutan för signering, alternativt rutan för avvikelse.

4.10.3 Expander

(EB, 2013-04-09, Forms v1.1.0.294)

Om man kopplar en trädnod till en Expander kommer den att automatiskt expanderas när användaren klickar på den noden i trädet. Dessutom kommer en eller flera nivåer av Expander expanderas om man klickar på en nod kopplat till ett element som ligger inuti dessa Expander.

För att expanderingen ska fungera optimalt så måste alla Expandrar finnas i trädet och vara placerade på i samma struktur som expandrarna i XAML koden, dock stör det inte i fall andra noder kopplade till andra element ligger emellan.

4.10.3.1 *XAML exempel*

```
<StackPanel>
  <Border>
    <StackPanel>
      <Expander x:Name="ExpanderSammanstallning">
        <XAML/>
      </Expander>
      <WrapPanel x:Name="InformationsText">
        <XAML/>
      </WrapPanel>
      <Expander x:Name="ExpanderPacemaker">
        <StackPanel>
          <Expander x:Name="ExpanderICDMaterial">
            <XAML/>
          </Expander>
          <Expander x:Name="ExpanderStyrkateter">
            <XAML/>
          </Expander>
          <XAML/>
          <Expander x:Name="ExpanderBiventrikular">
            <XAML/>
          </Expander>
          <XAML/>
        </StackPanel>
      </Expander>
    </StackPanel>
  </Broder>
</StackPanel>
```

4.10.3.2 *Motsvarande träd*

```
<RootHeaderItem Name1="BlankettNr"  Name2="BlankettNamn" Sync="Personnummer">
  <HeaderItem Name="Sammanställning" Sync="ExpanderSammanstallning">
    <Item Name="Notering" Sync="Notering" />
  </HeaderItem>
  <HeaderItem Name="Information" Sync="InformationsText"/>
  <HeaderItem Name="Pacemaker" Sync="ExpanderPacemaker">
    <Item Name="ICD material" Sync="ExpanderICDMaterial" />
    <Item Name="Styrkateter" Sync="ExpanderStyrkateter" />
    <Item Name="Biventrikulär" Sync="ExpanderBiventrikular" />
  </HeaderItem>
</RootHeaderItem>
```

4.11 Övriga fältfunktioner

Beskrivning av diverse övriga funktioner som kan användas på ett fält.

4.11.1 Automatiskt datum, med offset

Funktionen visar ett datum med fast offset. Nedanstående exempel visar hur man visar det datum som finns i optdatum med en offset på en dag(d1).

```
<TextBlock ToolTip="(Optdatum + 1)" Style="{DynamicResource textBlockTextBold}">
  <TextBlock.Tag>|opdatum|stringDateConverter|d1|xsDateRule|</TextBlock.Tag>
</TextBlock>
```

4.11.2Gömma/visa område efter kryssruta

(EB, 2011-02-21, Forms v1.1.0.178)

Lägg till följande kod överst (`StackPanel.Resources`) i XAML filen.

```
<!-- Used to handle Visibility.Collapsed (False) or Visibility.Visible(True) -->
<!-- Usage: Visibility="{Binding ElementName=CheckBoxOperation1, Path=IsChecked,
Converter={StaticResource BoolToVisConverter}}" -->

<BooleanToVisibilityConverter x:Key="BoolToVisConverter"/>
```

För element som ska gömmas/visas lägg till följande attribut:

```
Visibility="{Binding ElementName=CheckBoxOperation1, Path=IsChecked, Converter={StaticResource
BoolToVisConverter}}"
```

Elementet kommer då att vara synligt när kryssrutan är ikryssad och osynlig när den är urkryssad.

I dagsläget finns ingen möjlighet att göra det omvända, att visa om den kopplade kryssrutan är urkryssad och gömma när den är i kryssad. (ToDo: Testa) Eventuellt kan Tag-Convertern för BooleanVisibility användas till detta.

För att gömda fält ska kunna användas krävs att en variabel som Forms använder för detta definieras upp i StackPanel.Resources enligt:

```
<BooleanToVisibilityConverter x:Key="BoolToVisConverter"/>
```

På det objektet som ska gömmas används

```
Visibility="{Binding ElementName=CB1, Path=IsChecked,
Converter={StaticResource BoolToVisConverter}}"
```

där CB1 här är namnet på den kryssruta som ska styra

Detta fungerar endast mot kryssrutor, och det fungerar idag endast så att när kryssrutan är urkryssas så är objektet gömt och när rutan kryssas i så visas det.

Ska motsatsen göras får detta lösas med ett fusk genom att skapa två envalskryssrutor och förfylla ett värde i xml och ev. placera den ena så att den inte syns. Vill man göra det mot en ddl eller liknande så måste man också skapa kryssrutor som kopplas till samma xpath som ddl:en, placera kryssrutorna utom synhåll och koppla det som ska gömmas/visas mot kryssrutorna instället.

4.11.2.1 *Mot ComboBox*

Det finns för tillfället inget stöd för att i Forms använda sig av funktionen för att gömma fält mot en dropdownlist. Detta går dock att åstadkomma genom att först skapa en dropdownlist med de alternativ som ska finnas. Där efter för varje val lägga till en checkbox som alla kopplas mot samma xPath och där var och ens parameter sätts till det värde valet har i dropdownlisten. Detta betyder att när ett val i dropdownlisten välj så kommer motsvarande checkbox kryssas för i blanketten när den används i Forms. Man kan då koppla det som ska gömmas/visas utifrån varje val mot respektive checkbox. Till sist går det att gömma checkboxarna genom att placera dem utanför synbart område, förslagsvis med Canvas.Top/Left eller visibility=false.

4.11.3Låsa/låsa upp område efter kryssruta

Lägg till följande kod för element som ska låsas/låsas upp:

```
IsEnabled="{Binding ElementName=elem_signaturecontent_1, Path=IsEnabled}"
```

Motsvarande koppling går att göra mot om en kryssruta är urkryssad:

```
IsEnabled="{Binding ElementName=elem_signaturecontent_1, Path=IsChecked}"
```

I dagsläget är det inte möjligt att göra motsatserna till ovan, t.ex. att låsa om en CheckBox är ikryssad men det går i vissa fall lösa med hjälp av EnumBooleanConvertern.

Tänk på att IsEnabled låser alla underliggande element, inklusive Expander som inte går att expandera eller stänga och Länkar som inte kommer vara klickbara. För att undvika dessa måste alla element med liknande funktion kodmässigt placeras utanför de element som ska låsas.

4.12 ePen

(EB, 2012-11-14, Forms v1.1.0.285)

Funktionen för att hantera streck och tolkningar från DPoP kallas i Forms ePen.
För användaren består detta av en flik till vänster med följande:

- En informationsdel innehållande dockningsinformation och inställningsmöjligheter för hur tolkningar, streck och fält presenteras.
- En lista som innehåller rader med information om tolkningarna.

Streck som visas på blanketten.
Samt kopplingar mellan de olika delarna vid olika händelser.

4.12.1Förutsättningar

Alla `formitem2`s `elementid`:n ska vara antingen XAML fältets ElementTag.xPath. Tankar finns på att kunna använda x:Name istället och även om inte direkt stöd för det som elementid finns varken i Forms eller alla andra delar utanför så är Forms i alla fall skrivet med tanke på att detta skulle kunna läggas till.

Allt bygger på den komplexa typen `communicationform` som ser ut enligt följande, där varje `formitem2` under `formitems` ska vara tolkningen för ett fält i blanketten.

```
<communicationform dockid="{00000000-0000-0000-0000-000000000000}"
                   formid="-1"
                   dockversion="-1"
                   status="-1"
                   version="2010-09-17T15:45:48"
                   userid="">
  <formitems>
    <formitem2 gid="{00000000-0000-0000-0000-000000000000}"
               elementid="text"
               confirmed="1"
               confidence="100"
               starttime="2001-01-01T00:00:00"
               endtime="2001-01-01T00:00:00">
```

```
      <value/>
      <strokes/>
    </formitem2>
  </formitems>
</communicationform>
```

Forms anser att en tolkning är gjord om det finns data i värdet för Strokes. För att fungera måste också fältet finnas i blanketten annars kommer streck och tolkad data inte visas.

Den information som står i `communicationform` hanteras utanför Forms som själv bara läser från denna del av XML filen, med undantag för attributet `confirmed` som ändras av Forms om användaren gör ett val att godkänna eller ta bort godkännande från en tolkning.

4.12.2 Grundfunktion

Event från objekt i den grafiska vyn och funktioner direkt relaterade till initieringen och listan med tolkningar finns i `Page1`. Alla övriga funktioner för DPoP finns i `EPenHandler`. De flesta event som triggas i `Page1` gör bara grundläggande kontroller och anropar sedan funktionerna i `EPenHandler`.

Data för alla pennstreck och tolkningar hanteras i grunden av tre objekt i Forms.
`public Dictionary<FrameworkElement, UIElement> EPenHandler.EPenStrokeUIElementDictionary` - En lista innehållande alla fält i blanketten och tillhörande `Path` (pennstreck).

`public class EPenHandler.StrokeData` - Klass som hanterar data för en tolkning/fält/streck. Innehåller funktioner för att läsa upp (hämta), konvertera och tolka data på olika sätt.

`ListView Page1.ePenStrokeList.ItemsSource` - Listan innehållande alla tolkningar som visas för användaren. Listan innehåller all data som tillhandahålls av `StrokeData`, men bara en delmängd visas för användaren. Listan hålls också uppdaterad när ändringar görs så alla värden alltid ska vara aktuella.

4.12.3 Initiering

`private void SetupElementAddOnInfo(Panel panel)`
Vid uppstart kommer Forms, om blankettens XML innehåller `communicationform`, att gå igenom alla element i blanketten, kolla om de har en xPath och jämföra dem mot om varje `formitem2`.
För alla träffar skapas sedan en `Path` utifrån det data som står i `strokes`.
Om detta ger ett korrekt resultat (om data finns) kommer nuvarande element och den skapade `Path`:en läggas till i `EPenStrokeUIElementDictionary` och `Path`:en läggs även till i den `Canvas` som läggs till varje `FixedPage` för att hålla alla pennstreck och visa dem på blanketten i Formulär fliken.

`private void CreateEPenListView(String filter, String sort)`
Efter att alla streck/tolkningar har hittats skapas en `DataTable` som för varje element i `EPenStrokeUIElementDictionary` fylls på med data från fältets data i XML:et, `formitem2` samt fältet i sig. Denna data sätts sedan till `ePenStrokeList.ItemsSource` för att visas som tolkningsrader i listan.

4.12.4 Streck på blanketten

`public Path GeneratePath(FrameworkElement fwElem)`
Alla `Path` som skapas utifrån `strokes` placeras i en per sida gemensam `Canvas` som genereras och läggs till i respektive `FixedPage`.

`private Brush GetStrokePathColor(StrokeType strokeType)`
Varje streck får vid skapandet en färg utifrån deras typ och styr den färg de visas med på blanketten.

- Godkännda - `Brushes.Green`
- Felaktiga - `Brushes.Red`
- Utanför - `Brushes.Orange`
- Aktiv - `Brushes.Blue`
- Rättade - `Brushes.Gray`

4.12.5 Informationsdel

De värden och val som finnas under informationsdelen är:
Dockningens datum - Från XML:et

```
private void OnEPenFieldOpacityChanged(object sender,
RoutedPropertyChangedEventArgs<double> e)
```

`public String ChangeStrokeOpacity(Double opacity)`
Toning på streck och ifyllnadsfält (Slider med värden från 0-110). När värdet från `Slider`:n ändras går funktionen igenom alla streck och fält som har en tolkning och ändrar dess `Opacity`.

- 0-50 - Toning på Strokes där 0 = 0%, 50 = 100%
- 50-100 - Toning på Fält där 50 = 100%, 100 = 0%
- 100-110 - Tjocklek på Streck där >100 = 150%, >105 = 200%

Returnerar uträknat värde för streck och fält för att kunna visas för användaren.

```
private void OnEPenFieldConfidenceChanged(object sender,
RoutedPropertyChangedEventArgs<double> e)
```

`public void ChangeStrokeFieldConfidenceMarking(Double confidenceLevel)`
Markering av tolkningsgrad på fält (`Slider` med värden från 0-105). När värdet från `Slider`:n ändras går funktionen igenom alla fält som har en tolkning och ändrar dess `Background`.

- Fält med en tolkningsgrad < valt värde visas med `Brushes.Pink`.
- Fält med en tolkningsgrad >= valt värde visas med `Brushes.LightGreen`.
- Om värdet är 0 visas alla fält med normal bakgrund (`ClearValue`).

```
private void OnEPenStrokeVisibilityChanged(object sender, RoutedEventArgs e)
public void ShowStrokePaths(StrokeType display)
```

Visning av Strokes (baseras på `EPenHandler.StrokeType`)

- Visa Alla
- Visa Inga
- Visa Godkännda
  `private Boolean IsStrokeConfirmed(String fieldValue, String strokeValue, String confirmed)`
  De där värdet i fältet är lika med tolkningen eller `confirmed` är 1.
- Visa Felaktiga - De som är streck men inte är Godkänna, Utanför eller Rättade.

- Visa Utanför
  `private Boolean IsStrokeNotConnected(String ePenElementId)`
  Fältets xPath (skulle också kunna vara x:Name) innehåller orden `"PenStrokes"` och `"Page"`.
- Visa Rättade
  `private Boolean IsStrokeAdjusted(String fieldValue, String strokeValue)`
  Fältet innehåller ett värde som inte är lika med tolkningen och inte är tomt.

4.12.6 Tolkningsrader

4.12.6.1 *Kolumner*

Listan med tolkningsrader består av följande kolumner som får sitt värde från:

- Godkänd tolkning (Ok, kryssruta) - Från `confirmed` i XML:et.
- Tolkningsgrad (Grad, 0-100) - Från `confidence` i XML:et.
- Värdets namn (Namn, text) - Fältets ToolTip om denna inte är tom annars dess x:Name.
- Tolkat värde (Tolkning, text) - Från `value` i XML:et.
- Rättat värde (Rättning, text) - Fältets värde om tolkningen är av typen Rättade, annars tomt.

```
private Boolean IsStrokeEnabled(StrokeType strokeType)
```

Kryssrutan för godkänd tolkning är möjligt för användaren att ändra om tolkningen är av typen Godkänd, Felaktig eller Rättad.

1.1.1.1 Val av tolkningsrad eller val av fält

När man klickar på en tolkningsrad inträffar följande:

```
private void OnEPenStrokeListSelectionChanged(object sender, SelectionChangedEventArgs
e)
```

Här ges tillhörande ifyllnadsfält focus och det som ska inträffa därefter sköts av det event som inträffar när ett fält får focus. Detta för att samma saker ska inträffa oberoende om man väljer ett fält eller ett värde i tolkningslistan och för att se till att dessa endast anropas en gång. Det är dock viktigt att focus på fälten hanteras korrekt här då det annars finns risk för oändliga loopar annars.

```
void UIElement_GotFocus(object sender, RoutedEventArgs e)
public Boolean StrokeSelected(FrameworkElement selectedStrokeField)
```

När ett ifyllnadsfält får focus återställs först det som gjorts till föregående valda ifyllnadsfältet `lastSelectedStrokeField` och dess tillhörande `Path`. Därefter sätts det nu valda fältet till `lastSelectedStrokeField` för att kunna återställas nästa gång. Den `Path` som tillhör nuvarande fält får ny position, flyttas uppåt i förhållande till fältets storlek för att inte täcka fältet och får en ny färg, allt för att de valda strecken synas bättre och att det ska vara lättare för användaren att skriva i fältet.

```
private void SetSelectedStrokeBackground(FrameworkElement strokeField, Path
strokePath)
```

I samband med att en tolkning väljs placeras dessutom en bakgrund, i form av en `Border`, bakom den `Path` som tillhör tolkningen.

4.12.6.2 *Godkännande av tolkning*

När användaren godkänner en tolkning ska följande inträffa:

- Värdet ska hämtas från Communicationform och flyttas till blankettXML:et (eller till fältet, då detta betyder att värdet valideras?)
- Värdet ska markeras som Godkänt i Communicationform
- Tolkningsraden ska flyttas först nästa gång en uppdatering av tolkningslistan görs
- Fältet ska få focus. Detta är delvis en praktisk funktion då man alltid bör ge fältet focus om man ändrar på dess värde, bl.a. för att uträkningsfunktioner ska göras.

Flyttande av ram fungerar enbart om fältet ligger i en FixedPage eller Canvas.

4.13 ePrint

(EB, 2011-06-13, Forms v1.1.0.178)

Om konfigurering för detta är gjord lagras resultat XPS under:
U:\DDM_UTV\EyeDoc\AddOns\eBrev\ED_eBrevServer.service.net\App_Data där ”U\DDM_UTV\ EyeDoc” varierar beroende på viken server och miljö som testet gjorts i.

4.13.1XAML

För att kunna skicka eBrev läggs följande till i xaml-filen under formSettings:

```
<ebrev enabled="true" consignmentpath="blankett/consignment_1"/>
```

I XAML filen finns 3st krav som ska uppfyllas för att eBrevet ska fungera rätt. Formsettings, Quietzones, postnr och postort. Dessutom så måste fullständiga avsändar- och mottagaradresser finnas på rätt plats enligt postens regler, finns detta ej tillkommer en straffavgift till det skickade brevet.

Måndag tisdag onsdag, ska inte vara i mellanslag, det skiljer lite mellan våra och postens fonter. Därför får mellanslag inte användas för att skapa mellanrum, detta kan ofta finnas med i xps:er och måste plockas bort i XAML för ePrint blanketter för att kunna vara säker på att det skickade brevet kommer ha samma utseende som i Forms.

Man kan inte använda några färger i ePrint, detta eftersom ingen information om färger skickas i PREAFP filen från EyeDoc till Posten.

4.13.1.1 *Fonter*

För att kunna skicka brev med EyeDoc ePrint krävs det att alla fonter i Formsblanketten endast använder de av Stråfors godkända standard fonterna. I princip är detta Arial (stylesimulation, B, I), Times New Roman (stylesimulation, B, I, BI), Calibri och Verdana i storlekarna 8-12, 14, 16, 18, 20 & 24 samt Symbol och Code 128 i valfria storlekar. En komplett lista över alla finns under P:\Projekt\101 VGR\Projekt\eBrev\Arbetsmaterial\Integration\Posten\approvedfonts.xml. Detta är dock något som teoretiskt sett kan variera från avtal till avtal för olika kunder.

De storlekar som används i XAML för fonter är dessutom skalade med en faktor av 96/72 = 1.333333, vilket betyder t.ex. att storlek 12 ska skrivas 16 i XAML och 8 blir 10.6667 osv. Det ska dock noteras att xps-skrivaren (i alla fall i Windows XP) inte konverterar exakt enligt denna formel, men då en viss felmarginal tillåts så fungerar Glyphs direkt från en XPS även om t.ex. 8 där konverterats till 10.7204.

4.13.1.1.1 Storlekstabell

pt	px
6	8
8	10.6666
9	12
10	13.3333
11	14.6666
12	16
14	18.6666
16	21.3333
18	24
20	26.6666

4.13.2 Xml & Xsd

Samt följande i schemat:

```
<complexType name="Consignment_1">
  <sequence>
    <element name="consignmentid" type="tns:guid" minOccurs="0"/>
    <element name="timestamp" type="dateTime" minOccurs="0"/>
    <element name="internalstatusid" type="int" minOccurs="0"/>
    <element name="internalstatusname" type="tns:String50" minOccurs="0"/>
    <element name="postalcodexpath" type="tns:String50" minOccurs="0"/>
  </sequence>
</complexType>
<complexType name="Blankett">
  <sequence>
    <element name="consignment_1" type="tns:Consignment_1" minOccurs="0" maxOccurs="1"/>
  </sequence>
</complexType>
<element name="blankett" type="tns:Blankett"/>
```

Samt följande i xml filen:

```
<consignment_1>
  <consignmentid>00000000-0000-0000-0000-000000000000</consignmentid>
  <timestamp/>
  <internalstatusid/>
  <internalstatusname/>
  <postalcodexpath>/blankett/”XPath till element med postnummer”</postalcodexpath>
</consignment_1>
```

Notera att noden consignmentid har ett guid, utan måsvingar. Definitionen av typen kräver dock det.

Viktigt är också att i XML filen måste xPath:en till fältet för mottagarens postnummer finnas under blankett/consignment_1/postalcodexpath i format enligt t.ex. <postalcodexpath>/blankett/patient/adr/postn</postalcodexpath>.

Sedan ska också blankettens artikelnummer uppdateras under blankett/consignment_1/consignment_params/consignment_param[@typ='artikelnr']/value enligt

<consignment_param typ="artikelnr">

<value>Template_eBrev</value>

</consignment_param>

4.14 Skicka Intyg

4.15 Bilagor

(EB, 2011-02-21, Forms v1.1.0.178)

Bilagor måste vara xps:er, för att kunna användas som bilaga i Forms.

För att kunna hantera bilagor inkluderas olika delar i schema, initierings xml samt i xaml filen.

Infoga följande i xaml-filen under formSettings:

```
<attachment application="1" apparg="203" appver="1"/>
```

Samt följande i schemafilen

```
<!-- Attachment start -->
<simpleType name="enumAttachmentAppArg">
  <restriction base="integer">
    <enumeration value="201">
      <annotation>
        <documentation xml:lang="sv-SE">PM, XPS.</documentation>
      </annotation>
    </enumeration>
    <enumeration value="202">
      <annotation>
        <documentation xml:lang="sv-SE">Infoblad, XPS</documentation>
      </annotation>
    </enumeration>
    <enumeration value="203">
      <annotation>
        <documentation xml:lang="sv-SE">Bilaga, XPS</documentation>
      </annotation>
    </enumeration>
    <enumeration value="220">
      <annotation>
        <documentation xml:lang="sv-SE">Bilaga, eBrev</documentation>
```

```
      </annotation>
    </enumeration>
  </restriction>
</simpleType>
<simpleType name="enumAttachementApplication">
  <restriction base="integer">
    <enumeration value="1">
      <annotation>
        <documentation>archive</documentation>
        <documentation xml:lang="sv-SE">arkiv</documentation>
      </annotation>
    </enumeration>
    <enumeration value="2">
      <annotation>
        <documentation>content service</documentation>
        <documentation xml:lang="sv-SE">innehållstjänst</documentation>
      </annotation>
    </enumeration>
  </restriction>
</simpleType>
<complexType name="AttAddress">
  <annotation>
    <documentation>Defines what type of id to use.</documentation>
  </annotation>
  <sequence>
    <element name="value" type="tns:String50" minOccurs="0">
      <annotation>
        <documentation>Dummy node, to hold the value</documentation>
      </annotation>
    </element>
  </sequence>
  <attribute name="name" type="tns:String50">
    <annotation>
      <documentation>Application specific.</documentation>
    </annotation>
  </attribute>
  <attribute name="type" type="tns:String50">
    <annotation>
      <documentation>Schema types</documentation>
    </annotation>
  </attribute>
</complexType>
<complexType name="AttDocument">
  <annotation>
    <documentation>Field name is required for all enumAttachmentAppArg.</documentation>
  </annotation>
  <sequence>
    <element name="name" type="tns:String50" minOccurs="1"/>
    <element name="docid" type="tns:guid" minOccurs="0"/>
    <element name="attaddress" type="tns:AttAddress" minOccurs="0"/>
    <element name="attversion" type="tns:AttAddress" minOccurs="0"/>
    <element name="attargument" type="tns:AttAddress" minOccurs="0"/>
  </sequence>
  <attribute name="application" type="tns:enumAttachementApplication"/>
  <attribute name="apparg" type="tns:enumAttachmentAppArg"/>
</complexType>
<complexType name="Attachments">
  <annotation>
    <documentation>Holds a number of attachment's</documentation>
  </annotation>
  <sequence>
```

```
      <element name="attdocument" type="tns:AttDocument" minOccurs="0"
maxOccurs="unbounded"/>
    </sequence>
  </complexType>
  <!-- Attachment end -->
  <complexType name="Blankett">
    <sequence>
      <element name="attachments" type="tns:Attachments" minOccurs="0" maxOccurs="1"/>
    </sequence>
  </complexType>
  <element name="blankett" type="tns:Blankett"/>
```

Samt följande i xml-filen

```
<attachments>
  <attdocument application="0" apparg="0"><name/><docid/></attdocument>
  <attdocument application="0" apparg="0"><name/><docid/></attdocument>
  <attdocument application="0" apparg="0"><name/><docid/></attdocument>
  <attdocument application="0" apparg="0"><name/><docid/></attdocument>
  <attdocument application="0" apparg="0"><name/><docid/></attdocument>
  <attdocument application="0" apparg="0"><name/><docid/></attdocument>
  <attdocument application="0" apparg="0"><name/><docid/></attdocument>
  <attdocument application="0" apparg="0"><name/><docid/></attdocument>
  <attdocument application="0" apparg="0"><name/><docid/></attdocument>
  <attdocument application="0" apparg="0"><name/><docid/></attdocument>
</attachments>
```

4.15.1ePrint

För att användas som bilaga som ska skickas med ett eBrev så måste bilagan följa samma regler som brevet, med marginaler, typsnitt, storlekar osv. XPS:en måste också förberedas med ett tvätt verktyg (Z:\Program\xps\EyeDoc.Debug.Xps.XpsHandler\EyeDoc.Debug.Xps.XpsHandler.exe). Denna tvätt genomförs också när man laddar upp en XPS som en addhoc-bilaga i Forms.

Man kan också göra så att man importerar layouten in i en XAML fil som man publicerar till Archive som en Formsblankett och ser till så att den går att skicka som ett eBrev (postnummer måste finnas i XML:et och Formsettings inställningar måste finnas i XAML). Den Forms man använder sig av måste vara konfigurerad för att spara XPS:er till fil för ePrint. Skickar man sedan blanketten som eBrev så kommer man kunna plocka en XPS från den angivna platsen som går att användas som en ePrint-bilaga. Fördelen med detta är att man kommer kunna göra mindre justeringar av bilagan på ett mer kontrollerat sätt och har bättre kontroll på bilagans kvalitet.

4.16 PM

(EB, 2011-02-21, Forms v1.1.0.178)

Ett PM måste alltid vara en xps. Denna xps läggs upp i en blankett i Archive under fliken Dokument och där under som typen PM. PM-blankettens artikelnummer används som id för att PM-funktionen ska hitta rätt PM och får inte innehålla understräck eftersom det används som separator för knappen i XAML. Dessutom artikelnumret inte heller innehålla mellanslag utan bör endast innehålla a-z, A-Z.

I XML används denna komplexa typ:

```
<docholder>
    <doc type="0">
        <version>1</version>
        <name>PMname</name>
        <id>1</id>
        <url/>
    </doc>
</docholder>
```

`version` är okänt vad den har för funktion, låt den vara 1.
`name` är det artikelnummer som blanketten har i Archive som innehåller PM:et och måste vara exakt lika med det namnet för att fungera.
`id` kan användas för att kunna öppna och spara flera exemplar av samma PM. Det är oklart vad tanken och nyttan med den funktionen skulle vara. Det påverkar inte andra PM utan samma `id` (t.ex. 1) kan användas för olika `name`, så låt den vara 1.
`url` har troligen varit tänkt att användas för att skapa länkar, men det är både osäkert om den funktionen fungerar och bättre, säkrare, smidigare, stabilare att använda en `Hyperlink` i XAML istället hur som helst.

I XAML används det ända som behövs en knapp med rätt namn. När användaren sedan klickar på denna knapp i Forms så kommer PM:et att hämtas och visas i en flik.

```
<Button x:Name="docbtn_PMname_1"/>
```

I `x:Name` ska alltid `docbtn` stå först för att Forms ska veta att knappen är kopplad till ett PM, `_` separerar till `PMname` som är PM-blankettens artikelnummer i XML och Archive, `_` separerar till `1` som är kopplat till `id` i XML.

Att skapa xps:er som ser bra ut i Forms är däremot svårt. pga den sandbox som används i Forms så kan Forms inte använda sig av en i .NET inbyggda xpsläsaren utan Forms har en inbyggd som inte är lika bra på att läsa xpser.

I Forms för EyeDoc 2.0 får xps:erna som ska vara PM inte heller innehålla bilder (ej vektoriserade resurser i xps:en), detta fungerar däremot i EyeDoc 3.0+.

4.17 Versionshantering

4.17.1Visa versioner

När en blankett sparas första gången i en ny session, skapas en version. En ny ”extra” version skapas dessutom vid spara till signeringskorg. Innehåller blanketten flera versioner kan tidigare versioner öppnas i skrivskyddat läge, se fliken, Ärende.

4.17.2 Versionshantering av PM baserat på versioner av blankett

Versionshantering av PM baserat på versioner av blankett, innebär att man låser aktuella PM till blanketten i samband med att användaren trycker på ”Spara” och stänger EyeDoc Forms. (Detta gäller inte för de PM som ligger innanför signeringsblock). Varje gång användaren klickar på ”Spara” och stänger EyeDoc Forms genereras det en ny version av SVP, och gamla versioner är låsta för ifyllnad. I samband ”Spara” låses aktuell version av PM till aktuell blankett. Detta innebär att när användaren åter öppnar en sparad blankett för ifyllnad kommer den senaste versionen av aktuellt PM att vara tillgänglig, men sparas (knyts) inte till aktuell blankett förrän användaren klickar på ”Spara”. D.v.s. om användaren läser ett nytt PM och sedan klickar på ”Avbryt” så knyts inte detta PM till blanketten. Fördelen med denna lösning är att de PM som ligger utanför fält med ifyllnad och signering kontinuerligt versionshanteras samtidigt som den senaste versionen alltid visas i arbetsversionen.

4.18 Math

(EB, 2012-08-01, Forms v1.1.0.275)

Math tillför beräkningsfunktionallitet till en blankett och fungerar på alla typer av fält som stödjer uppdatering av Forms blankett-XML-fil.

Aktiveringen av Math samt konfigureringen av funktionerna görs i XAML-filens befintliga Formsettings:

```
<formsettings settingversion="1" formversion="1">

  <Math active="1">
    <calculatorsetting name="calc1" >
      <source>xPath1;xPath;xPath3;xPath4</source>
      <formel>{0} * ({1} + {2}) - {3}</formel>
      <triggerxpath>xPath4</triggerxpath>
      <resultxpath>xPath5</resultxpath>
    </calculatorsetting>
    <calculatorsetting name="calc2" >
      <source>xPath5;xPath6</source>
      <formel>{0} + {1}</formel>
      <triggerxpath>xPath6</triggerxpath>
      <resultxpath>xPath7</resultxpath>
    </calculatorsetting>
  </Math>

</formsettings>
```

Attributet `active` i `Math` elementet måste sättas till 1 för att beräkningsfunktioner ska vara aktiverade i Forms och kan då inaktiveras genom att sätta detta till 0. Math stödjer att flera formler används i en blankett, men varje formel ska ha ett eget unik namn t.ex. `name="calc1"`. Under varje `calculatorsetting` ska det finnas 4 underelement som definierar funktionaliteten för beräkningsfunktionen.

`source` = Lista med xPath till de värden i XML:et som skall användas som invärden i funktionen. Varje xPath separeras med ett semikolon, men inget före den första eller efter den sista.

formel = Det matematiska uttryck eller den Math funktion som ska användas. Värdena från Source används genom att skriva {0} är första, {1} för andra osv. De matematiska uttryck som stödjs är (), +, -, /, *. För övriga Math funktioner se kommande underkapitel.
triggerxpath = Är en unik nyckel för den specifika beräkningsfunktionen. Den måste inte vara i xpath syntax utan kan vara som fri text. 2 olika calculatorsettings kan inte ha samma text.
resultxpath = Den xPath dit resultatet av uträkningen ska sparas.

Ett resultat från en uträkning kan användas som invärde i en annan formel. Se source i calc2 ovan där den tar in xPath5 som invärde vilken även finns som resultxpath i calc1. Notera också att om en resultxpath för en funktion finns i source för en annan så kommer också uträkning av denna triggas. Sedan bör inte flera olika beräkningsfunktioner ha samma resultxpath även om detta är möjligt då de skriver över varandras värden och logiska fel i flödet kan uppstå.

4.18.1 Design

Kräver att elementet har Focus.
Körs vid xml change.

4.18.2 Math IF

Jämför ett värde med ett påstående eller i ett intervall och ger ett värde utifrån resultatet av denna jämförelse.

```
<formel>[IF|Value|Min|Function|Max|IfLow|IfTrue|IfHigh]</formel>
```

IF = Namnet på funktionen.
Value = Det värde som ska jämföras. Numeriskt värde eller Math funktion.
Min = Det lägre värdet i en jämförelse mot ett intervall. Numeriskt värde eller Math funktion. Används enbart för intervall.
Function = Den logiska funktion som skall användas:

L	Value < Max
G	Value > Max
==	Value == Max
L=	Value <= Max
G=	Value >= Max
!=	Value != Max
LL	Min < Value < Max
L=L	Min <= Value < Max
LL=	Min < Value <= Max
L=L=	Min <= Value <= Max

Max = Värde att alltid jämföra mot. Numeriskt värde eller Math funktion. Används även som det högre värdet i intervall.
IfLow = Det resultat som ges om Value ligger till vänster om intervallet (om Value är mindre än alt. lika med Min).
IfTrue = Det resultat som ges om resultatet av jämförelsen är sant eller om Value ligger inom intervallet.

`IfHigh` = Det resultat som ges om resultatet av jämförelsen är falskt eller om Value ligger till höger om intervallet (är större än alt. lika med Max).
För att fungera måste alla uppgifter anges korrekt, inga parametrar får utelämnas, alla parametrar är skiftlägeskänsliga. Stödjer i dagslägen inte någon kombination med eventuella andra Math funktioner och formeln måste skrivas exakt enligt beskrivningen vilket inkluderar antalet pipetecken och hakparenteser som är reserverade tecken.
Exempel:

```
<formel>[IF|{0}+{1}|5|LL|12|{2}|15|{3}]</formel>
```

Om {2} = 10 och {3} = 20
då {0} = 4 och {1} = 5 kommer resultatet att bli 15
då {0} = 1 och {1} = 3 kommer resultatet att bli 10
då {0} = 14 och {1} = 2 kommer resultatet att bli 20

4.18.3 Math ROUND

Avrundar ett tal till ett antal decimaler. Denna funktion avrundar det faktiska matematiska värdet, inte nödvändigtvis till antalet decimaler som visas, t.ex. så kan värdet 2.01 avrundas till 2.0 för både 0 och 1 decimaler.

```
<formel>[ROUND|Value|Decimal|Type]</formel>
```

`ROUND` = Namnet på funktionen.
`Value` = Det värde som ska avrundas. Numeriskt värde eller Math funktion (t.ex. {1}*2).
`Decimal` = Heltal för antalet decimaler som ska avrundas till.
`Type` = Vilken typ av avrundning som används:

- `ToEven` = Avrundar till närmaste heltal (t.ex. 0.5 = 0, 1.5 = 2).
- `AwayFromZero` = Avrundar alltid uppåt (t.ex. 0.5 = 1, 1.5 = 2).

För att fungera måste alla uppgifter anges korrekt, inga parametrar får utelämnas, alla parametrar är skiftlägeskänsliga. Stödjer i dagslägen inte någon kombination med eventuella andra Math funktioner och formeln måste skrivas exakt enligt beskrivningen vilket inkluderar antalet pipetecken och hakparenteser som är reserverade tecken.
Exempel:

```
<formel>[ROUND|{0}+{1}|1|ToEven]</formel>
```

Om {0} = 1.2 och {1} = 0.47 så kommer resultatet att bli 1.7.

4.19 Validering

(EB, 2015-08-06, Forms v1.1.0.344)

Dessa valideringsregler är en utökning och påbyggnad av de grundläggande regler som finns i Rule. Detta möjliggör både mer avancerade funktion och möjligheten till att ha flera regler på varje fält. Till skillnad från Converter och Rule så kommer ExtendedRule inte att radera felaktiga värden.

Definitionen görs i första hand på ExtendedRule, men vissa regler använder sig av ytterligare inställningar i Formsettings, vilket i så fall står angivet i kapitlet för respektive regel.

Valideringarna görs via XML-värdet för fältet som sparas ut av fältets Converter. Detta betyder att till skillnad från äldre versioner så fungerar samtliga valideringsregler på alla typer av fält som kan spara värden till XML-filen om inget annat anges för respektive regel.

Om ett valideringsfel uppstår kommer angivet felmeddelandet kommer användas som inforuta för fältet och kommer markeras med en valideringsmarkering med en ikon beroende på angiven Valideringsnivå. Om flera valideringsfel finns på samma fält kommer det vara det första felet av den högsta Valideringsnivån som är den som kommer att visas.

I samband med att man sparar och utför vissa andra funktioner som t.ex. delsignering så kommer en fullständig validering göras och resultatet visas då i en valideringsdialog som listar samtliga unika valideringsfelmeddelanden.

Valideringen i sig körs först när man öppnar en blankett för att från början visa eventuella valideringsmarkeringar. Därefter körs validering på samtliga fält när användaren gör en ändring på ett fält för att regler som eventuellt är beroende av det ändrade fältet. Dessutom så körs fältets Rule också igenom i samband med att de utökade valideringsreglerna körs så också dess eventuella felmeddelande kommer med i valideringsdialogen.

Det går att kombinera olika utökade valideringsregler fritt utan någon faktisk begränsning på hur många som kan användas på varje fält. Dock behöver man se till så att dessa varken motsäger varandra eller angiven Rule.

4.19.1Valideringsnivå

(EB, 2014-04-12, Forms v1.1.0.331)

Följande är de nivåer som kan användas för en valideringsregel på ett fält (första parametern i regelns tag):

Namn	Rank	Ikon	Betydelse
Fatal	100	!	Allvarligt fel som måste åtgärdas direkt.
Critical	90	!	Allvarligt fel.
Important	80	!	Allvarligt fel.
Exception	70	#	Fel i blankettens inställningar eller i Forms.
Binding	60	!	Felaktig datatyp på inmatat värde.
Mandatory	50	*	Fel som måste åtgärdas.
Warning	40	?	Fel som bör åtgärdas.
Recommended	30	i	Notering som behöver uppmärksammas, som kanske är ett fel eller inte.
Noteworty	20	i	Notering som enbart behöver visas i blanketten.

Notera att värdet på "Rank" kommer kunna ändras, dock lär inte ordningen ändras. Betydelsen av fel av respektive nivå går att ändra från blanketten se Formsettings xxx.

Följande är nivåer som i Forms indikerar nivåer som ska ignoreras:

Namn	Rank	Ikon	Betydelse
Disabled	500	-	Betyder att fel enbart ska visas om dess nivå är större än Disabled, vilket betyder aldrig.
None	0	-	Betyder att nuvarande fält inte har något valideringsfel.

Dessa kan inte användas på valideringsregler.
Om None används kommer det betyda att regeln inte ger något fel alls.
Om Disabled används kommer betyda den högsta nivån av fel, högre än Fatal.

Följande är funktionsspecifika nivåer:

Namn	Rank	Ikon
Validate	7	i
Open	6	i
Save	5	i
Done	4	i
DonePreSign	3	i
SignLock	2	i
SingleSign	1	i

Alla dessa ska enbart användas på valideringsregeln i fältet tag attribut, ej som en nivå i formsettings (då dessa är lägst och kommer trigga alla nivåer över).
Här har värdet på "Rank" ingen betydelse, dessa ligger praktiskt sätt på samma nivå, men kan ej sättas upp så då varje värde måste vara unikt för att funktionsspecifika valideringsfel enbart ska visas i den funktionens dialog.

Samtliga kommer visa valideringsmarkeringar i blanketten, men i dialogerna kommer de enbart visas om de konfigurerats att göra det och då enbart de som tillhör den aktuella funktionen.

4.19.2 Dialogtyper

(EB, 2014-04-12, Forms v1.1.0.331)

De stora ikoner som visas till vänster i dialogföntret:

Nivå	Ikon	Används som standard av dialogtyp
Error		Block, Error
Warning		Warning
Question		-
Info		Info

Dialogikoner och knappar i dialogerna kan inte konfigureras utan är fasta per dialogtyp och funktionstyp.

4.19.3 Valideringsikoner

(EB, 2014-04-12, Forms v1.1.0.331)

I dialogerna finns också 3 valideringsnivåer som styr färgen på ikonerna framför valideringsfelet:

Nivå	Färg	Betydelse
Red	Röd	Error - Felet blockerar funktionen och måste rättas.
Yellow	Gul	Warning - Felet är viktigt, men tillåter funktionen att köras.
Blue	Blå	Info - Felet är inte viktigt, men bör uppmärksammas.

Fel av en lägre nivå än Blue visas inte i dialogerna, utan enbart i blanketten.

Notera att betydelsen kan ändras beroende på konfigurationen i blanketten, men konfigurationen bör å andra sidan göras så att dessa betydelser följs.

4.19.4 Funktionstyper

(EB, 2014-04-12, Forms v1.1.0.331)

Följande beskriver vilka funktion som kan visas valideringdialoger och vilka valideringsnivåer som triggar vilken dialogtyp för respektive funktion som standard:

Namn	Funktion	Block	Error	Warn	Info
Validate	Validering	Disabled	Disabled	Disabled	Fatal
Open	Öppna blankett	Disabled	Disabled	Disabled	Important
Save	Spara	Fatal	Disabled	Critical	Important
Done	Spara klar	Fatal	Mandatory	Recommended	Disabled
DonePreSign	Autospara vid signering	Warning	Disabled	Disabled	Disabled
SignLock	Signering	Warning	Disabled	Recommended	Disabled
SingleSign	Delsignering	Warning	Disabled	Disabled	Disabled

Understruken text indikerar att en standarddialog finns som för Error och Warning. Övriga kan aktiveras från blanketten och då kommer Block, Error och Warning visa en OK-dialog och stoppa funktionen medan Info kommer visa en OK-dialog men låta funktionen köra.

Alla inställningar och vilka valideringsfel som får vilken färg går att ställa in från blanketten.
Referens: Se kapitel 4.5.3 Validation för beskrivning av inställningsmöjligheterna respektive 4.5.3.3 Standardinställningar för vilka inställningar som används om inget annat anges.

4.19.4.1 *BitStatus - Spara ofullständigt ifylld*

(EB, 2014-04-12, Forms v1.1.0.331)

Vilken bitStatus som sätts vid spara går inte att konfigurera utan sätts hårt och på samma sätt för både arbetsdokument (status 10) och klar (status 20). Detta bör man ta hänsyn till innan inställningar ändras för funktionerna spara (Save) och spara klar (Done).

Valideringsnivå	BitStatus
<= Recommended	48
= Warning	16 (inte 32)
>= Mandatory	0 (inte 16, inte 32)

Notera att detta betyder att en blankett som innehåller valideringsfel har BitStatus ”0” (inte 16, inte 32), en blankett som inte har någon validering på sig har exakt samma status så det går inte att skilja på dessa. Egentligen borde "16" betyda "Blanketten har valideringsfunktioner" och "32" betyda "Validering utförts OK".

Detta används idag enbart för de vanliga sparaknapparna (saveDestination 1 & 2) och inte för eBrev (saveDestination 17). eBrev får därför alltid BitStatus ”0” (inte 16, inte 32).

4.19.5 Funktionsspecifika nivåer

(EB, 2014-04-12, Forms v1.1.0.331)

Funktionsspecifika nivåer innebär valideringsregler av en nivå som enbart ska visas för en specifik funktion (**Referens:** Se kapitel 4.19.4 Funktionstyper).
Detta är i första hand användbar vid öppnade av blankett (Funktionstyp Open), men kan användas för samtliga Funktionstyper.

Färgen på ikoner för valideringsfel (om de visas) styrs av vilken dialogtyp som har konfigurerats. Anledningen till att funktionsdialogen använder sig av en av de andra dialogtyperna är för att man ska kunna placera den i rätt nivå, t.ex. om man vill att den ska visas först kan man använda Block.

4.19.6 Valideringsfunktioner

(EB, 2014-04-13, Forms v1.1.0.331)

Normalt körs valideringen enbart om fältet har ett värde är angett. Undantag får detta är så klart samtliga Mandatory regler som validerar att fältet har ett värde angett. Vill man därför att en regel ska validera både värdet i sig och att det är ifyllt kombinerar man flera passande regler med varandra.

Referens: För information om hur valideringsreglernas uppbyggnad i taggen se respektive kapitel nedan och för generell information om deras placering i fältets Tag attribut se kapitel 4.6.6 ExtendedRule.

Följande är de funktioner som finns tillgängliga för användning i blanketter i Forms:

Valideringsregel	Beskrivning
Mandatory	Ett värde måste anges i fältet, det får inte lämnas tomt.
DependentMandatory	Ett värde måste anges i fältet om ett annat fält är ifyllt, i så fall får det inte lämnas tomt.

WarningMandatory	Samma funktion som Mandatory.
MinMax	Värdet i fältet måste vara ett flyttal inom ett specifikt intervall, förutsatt att ett värde har angetts.
RegExp	Värdet i fältet måste följa ett specifikt format, förutsatt att ett värde har angetts.
PosNeg	Värdet i fältet måste vara ett antingen negativt eller positivt tal, förutsatt att ett värde har angetts.
DecimalFormat	Formateringsregel som visar ett specifikt antal decimaler för ett tal även om det faktiska värdet består av ett annat antal decimaler, förutsatt att ett värde har angetts. Fungerar enbart för TextBox och TextBlock.
StartEndDate	Värdet i fältet måste vara ett datum inom ett specifikt intervall eller i förhållande till ett datum i ett annat fält, förutsatt att ett värde har angetts.
GroupDependantMandatory	Ett värde måste anges i varje av flera fält, i vissa fall om ett annat fält har eller inte har ett värde angett.
GroupChoiceDependantMandatory	Ett värde måste anges i ett visst antal av flera fält, i vissa fall om ett annat fält har eller inte har ett värde angett.
StringFormat	Formateringsregel som visar ett angivet värde i ett specifikt format. Fungerar enbart för TextBox och TextBlock.
StringTooLarge	Formateringsregel som visar en ersättningstext för värden som är för stora för att få plats i fältet. Fungerar enbart för TextBox och TextBlock.
Int32	Värdet i fältet måste vara ett korrekt heltal, förutsatt att ett värde har angetts.
Double	Värdet i fältet måste vara ett korrekt flyttal, förutsatt att ett värde har angetts.
SystemValue	Värdet i fältet måste vara samma som ett metadatavärde i systemet, förutsatt att ett värde har angetts.
IdentificationNumber	Värdet i fältet måste följa formatet för ett specifikt id-nummer, t.ex. ett personnummer, förutsatt att ett värde har angetts.

Notera att Valideringsnivån kommer vara Mandatory om inget annat anges för regeln på fältet, med undantag från WarningMandatory som får Warning som standardnivå.

4.19.6.1 *Mandatory*

(EB, 2014-04-19, Forms v1.1.0.331)

Validerar att fältet inte lämnas utan ett värde. Kontrollerar enbart att ett värde finns så kombineras ofta med andra valideringsfunktioner för att också validera formatet/innehållet på värdet.

Följande parametrar finns att ange för regeln:

Parameter	Beskrivning
[0]	Felmeddelande som ska visas vid valideringsfel. Ersätts med en standardtext från språkfilen om det är tomt eller utelämnas.

Följande parametrar går att använda i felmeddelandetexten:

Strängparameter	Beskrivning
-	Används ej.

4.19.6.1.1 Exempel

```
<TextBox.Tag>
  Antal biljertter|xPath|stringInt32Converter||xsInt32Rule
  |Important;Mandatory;Du måste ange hur många biljetter du vill beställa.
</TextBox.Tag>
```

4.19.6.2 *DependentMandatory*

(EB, 2014-04-19, Forms v1.1.0.331)

Validerar att fältet inte får lämnas utan ett värde som ett annat beroende fält har ett värde. Denna regel stödjer inte att göra ett fält obligatoriskt om det beroende fältet inte är ifyllt, då får GroupDependantMandatory användas istället.

Följande parametrar finns att ange för regeln:

Parameter	Beskrivning
[0]	Namnet på det fält som gör fältet med regeln obligatorisk. Om parametern är tom, saknas eller om angivet fältnamn inte finns i blanketten kommer fältet göras obligatoriskt.
[1]	Felmeddelande som ska visas vid valideringsfel. Ersätts med en standardtext från språkfilen om det är tomt eller utelämnas.

Följande parametrar går att använda i felmeddelandetexten:

Strängparameter	Beskrivning
{0}	Det beroende fältets namn (ElementName).

4.19.6.2.1 Exempel

```
<CheckBox x:Name="ReasonsOther"/>
<TextBox x:Name="OtherReason">
```

```
  <TextBox.Tag>
    Annan orsak|xPath|stringEncodedConverter||
    |Warning;DependentMandatory;ReasonsOther:Eftersom du valt {0} måste du beskriva
vilken orsak som gäller.
  </TextBox.Tag>
</TextBox>
```

4.19.6.3 *WarningMandatory*

(EB, 2014-04-19, Forms v1.1.0.331)

Validerar att fältet inte lämnas utan ett värde. Har exakt samma funktion som Mandatory förutom att Valideringsnivån är Warning som standard istället för Mandatory om inget annat anges, **Referens:** Se kapitel: 4.19.6.1 Mandatory.

Följande parametrar finns att ange för regeln:

Parameter	Beskrivning
[0]	Felmeddelande som ska visas vid valideringsfel. Ersätts med en standardtext från språkfilen om det är tomt eller utelämnas.

Följande parametrar går att använda i felmeddelandetexten:

Strängparameter	Beskrivning
-	Används ej.

4.19.6.3.1 Exempel

```
<TextBox.Tag>
  Antal biljertter|xPath|stringInt32Converter||xsInt32Rule
  |;WarningMandatory;Du borde ange hur många biljetter du vill beställa.
</TextBox.Tag>
```

4.19.6.4 *MinMax*

(EB, 2014-04-19, Forms v1.1.0.331)

Validerar att värdet som anges är ett korrekt tal och att dess numeriska värde ligger inom angivet intervall.

Följande parametrar finns att ange för regeln:

Parameter	Beskrivning
[0]	Det numeriska tal som är det lägsta tillåtna talet som kan anges i fältet. Om parametern är tom, saknas eller inte är ett korrekt tal sätts ingen begränsning nedåt, **Referens:** Se Double.MinValue.
[1]	Det numeriska tal som är det högsta tillåtna talet som kan anges i fältet. Om parametern är tom, saknas eller inte är ett korrekt tal sätts ingen begränsning uppåt, **Referens:** Se Double.MaxValue.
[2]	Felmeddelande som ska visas vid valideringsfel. Ersätts med en standardtext från språkfilen om det är tomt eller utelämnas.

Följande parametrar går att använda i felmeddelandetexten:

Strängparameter	Beskrivning
{0}	Det angivna värdet i fältet.
{1}	Det lägsta tillåtna värdet, från parameter [0]
{2}	Det högsta tillåtna värdet, från parameter [1]

4.19.6.4.1 Exempel

```
<TextBox x:Name="Copies">
  <TextBox.Tag>
    Antal kopior|xPath|stringEncodedConverter||
    |Mandatory;MinMax;5:100:Du har valt {0} kopior, men måste beställa som minst {1}st
och som mest {2}st.
  </TextBox.Tag>
</TextBox>
```

4.19.6.5 *RegExp*

(EB, 2014-04-19, Forms v1.1.0.331)

Validerar att fältets värde följer ett format enligt angivet reguljärt uttryck.

Följande parametrar finns att ange för regeln:

Parameter	Beskrivning
[0]	Den reguljära uttryck som värdet i fältet måste följa. Måste börja med ”^” (cirkumflex) och avslutas med ”$” (dollar).
[1]	Felmeddelande som ska visas vid valideringsfel. Ersätts med en standardtext från språkfilen om det är tomt eller utelämnas.

Följande parametrar går att använda i felmeddelandetexten:

Strängparameter	Beskrivning
{0}	Det angivna värdet i fältet.
{1}	Angiven Regular Expression, från parameter [0].

Notera att reguljära uttryck ofta innehåller tecken som måste teckenkodas, t.ex är ”|” (vertikalstreck) ett tecken som ofta används. **Referens:** Se kapitel 4.6.7 Encoding och decoding.

4.19.6.5.1 Exempel

```
<TextBox.Tag>
  MasterCard|xPath|stringEncodedConverter||
  |Mandatory;RegExp^5[1-5][0-9]{14}$:Värdet {0} är inte ett korrekt MasterCard nummer.
</TextBox.Tag>
```

4.19.6.6 *PosNeg*

(EB, 2014-04-19, Forms v1.1.0.331)

Validerar att fältets värde är ett numeriskt tal och att det är antingen positivt eller negativt.

Följande parametrar finns att ange för regeln:

Parameter	Beskrivning
[0]	Ett heltal som anger om enbart positiva eller negativa tal ska tillåtas. Om parameterns värde är positivt godkänns enbart positiva värde i fältet och tvärt om. Om parametern är tom, saknas eller inte är ett korrekt heltal kommer ”0” användas vilket betyder att enbart positiva tal godkänns.
[1]	Felmeddelande som ska visas vid valideringsfel. Ersätts med en standardtext från språkfilen om det är tomt eller utelämnas.

Följande parametrar går att använda i felmeddelandetexten:

Strängparameter	Beskrivning
{0}	Det angivna värdet i fältet.
{1}	Texten för antingen positivt eller negativt som hämtas från språkfilen beroende på parametern [0].

Notera att ”0” (noll) räknas som positivt i både styrparametern ([0]) och värdet på fältet.

4.19.6.6.1 Exempel

```
<TextBox.Tag>
  Temperatur|xPath|stringEncodedConverter||
  |Mandatory;PosNeg;-1:Angivet värde ’{0}’ måste vara ett {1} tal.
</TextBox.Tag>
```

4.19.6.7 *DecimalFormat*

(EB, 2015-08-05, Forms v1.1.0.344)

DecimalFormat är en formateringsregel. Den validerar att fältets värde är ett korrekt flyttal och formaterar sedan talet med ett angivet antal decimaler. Notera att denna regel ändrar på det faktiska värdet som sparas.

Följande parametrar finns att ange för regeln:

Parameter	Beskrivning
[0]	Antalet decimaler som ska visas. Om parametern är tom, saknas eller inte är ett korrekt heltal används ”0” som antalet decimaler (noll/inga).
[1]	Felmeddelande som ska visas vid valideringsfel. Ersätts med en standardtext från språkfilen om det är tomt eller utelämnas.

Följande parametrar går att använda i felmeddelandetexten:

Strängparameter	Beskrivning
{0}	Det angivna värdet i fältet.

Notera att formateringen enbart fungerar för TextBox och TextBlock. På övriga fält kommer dock värdet valideras att det är ett korrekt flyttal, men då bör istället DoubleRule användas, **Referens:** Se kapitel 4.19.6.11 Double.

Avrundningar som används sker mot närmaste jämna tal i den sista siffran efter avrundning. Alltså blir 1.25 = 1.2, 0.35 = 0.4 och 6.5 = 6 för en, en, respektive noll decimaler.

4.19.6.7.1 Exempel

```
<TextBox.Tag>
  Antal|xPath|stringDoubleConverter||xsDoubleRule
  |Mandatory;DecimalFormat;2:Värdet {0} är inte ett korrekt flyttal.
</TextBox.Tag>
```

4.19.6.8 *StartEndDate*

(EB, 2014-04-19, Forms v1.1.0.331)

Validerar att fältet är ett korrekt datum och att datumet är inom en begränsad tid före eller efter ett annat datum. Datumet som ska jämföras emot kan vara ett värde från ett annat datumfält, ett fast datum eller dagens datum. Funktionen kan användas för både datum och tid.

Följande parametrar finns att ange för regeln:

Parameter	Beskrivning
[0]	Basdatumet som jämförelsen ska utgå ifrån. [Now] används för dagens datum, annars namnet på det fält som jämförelsen ska göras med, eller ett fast datum i formatet yyyy-MM-dd. Om parametern är tom eller om det beroende fältet inte har något värde görs ingen validering. Om angivet fält inte finns eller om parametern är på annat sätt felaktigt angiven ges ett valideringsfel av nivån Exception.
[1]	Tidsspann att lägga till basdatumet för jämförelsen av tidigaste godkända datum. För format se tabell nedan. Om parametern är tom eller saknas sätts ingen begränsning nedåt, om den är i ett felaktigt format ges ett valideringsfel av nivån Exception.
[2]	Tidsspann att lägga till basdatumet för jämförelsen av senaste godkända datum. För format se tabell nedan. Om parametern är tom eller saknas sätts ingen begränsning uppåt, om den är i ett felaktigt format ges ett valideringsfel av nivån Exception.
[3]	Felmeddelande som ska visas vid valideringsfel. Ersätts med en standardtext från språkfilen om det är tomt eller utelämnas.

Om blanketten är låst/skrivskyddad görs ingen validering mot dagens datum om ”[Now]” har angetts.

Följande parametrar går att använda i felmeddelandetexten:

Strängparameter	Beskrivning

{0}	Det angivna värdet i fältet.
{1}	Det tidigaste tillåtna värdet, från parameter [1].
{2}	Det senaste tillåtna värdet, från parameter [1].

Tidsspannsparametrarna anges enligt följande:

Parameter	Beskrivning
Y	År
M	Månad
D	Dag
H	Timma
I	Minut
S	Sekund

Varje bokstav måste versal och följas av ett positivt heltal. Olika tidstyper kan sättas samman med varandra och för att dra ifrån tid används ett ”-” (minus) framför serien. T.ex. -Y2M5 (minus två år och fem månader) eller M19D7 (19 månader och sju dagar).

4.19.6.8.1 Exempel

```
<TextBox x:Name="StartDate">
  <TextBox.Tag>
    Startdatum|xPath|stringEncodedConverter||
    |Mandatory;StartEndDate;[Now]:Y-1:Y1:Angivet datum {0} måste vara inom ett år från
dagens datum (mellan {1} och {2}).
  </TextBox.Tag>
</TextBox>
<TextBox x:Name="EndDate">
  <TextBox.Tag>
    Slutdatum|xPath|stringEncodedConverter||
    |Mandatory;StartEndDate;EndDate::Y1:Angivet datum {0} måste vara inom ett år efter
startdatumet.
  </TextBox.Tag>
</TextBox>
```

4.19.6.9 *StringFormat*

(EB, 2015-08-05, Forms v1.1.0.344)

StringFormat är en formateringsregel. Den formaterar en text utefter ett angivet format. Notera att denna regel ändrar på det faktiska värdet som sparas.

Följande parametrar finns att ange för regeln:

Parameter	Beskrivning
[0]	Det format som värdet ska formateras i, med hjälp av `string.Format`.
[1]	Felmeddelande som ska visas vid valideringsfel. Ersätts med en standardtext från språkfilen om det är tomt eller utelämnas.

Följande parametrar går att använda i felmeddelandetexten:

Strängparameter	Beskrivning
{0}	Det angivna värdet i fältet.
{1}	Det angivna formatet i parameter [0].

Patternsutformning måste börja med ”{” och sluta med ”}” och alla reserverade tecken måste teckenkodas. **Referens:** Se kapitel 4.6.7 Encoding och decoding.

Att tänka på också är att StringFormat kan förändra datatypen av ett värde från ett tal till en sträng, så rätt typ av konverter måste användas utifrån detta. **Referens:** Se kapitel 4.6.3 Converter.

4.19.6.9.1 Exempel

```
<TextBox.Tag>
  Telefonnummer|xPath|stringEncodedConverter||
  |Mandatory;StringFormat;{0:(###) ###-####}:Angivet värde {0} är inte ett korrekt
telefonnummer.
</TextBox.Tag>
```

Om användaren matar in texten 8005551212 i textboxen ovan blir resultatet efter formateringen (800) 555-1212.

4.19.6.9.2 Formatbeskrivning

Texten i klammerparenteser är ***{index[,alignment][:formatString]}***.Om anpassningen är positivt, är texten högerjusterad i ett fält av visst antal platser, om det är negativt, det är vänsterjusterad. Observera att om man använder StringFormat av ett tal så är resultat inte längre ett tal utan har formatet sträng vilket kan ge problem om fältet är specificerad till ett tal eller om den används in matte beräkningar etc.

Ex inmatning av texten Hello	Ger resultatet
{->{1,10}<-}	-> Hello<-
{->{1,-10}<-}	->Hello <-

Nummer

Grundläggande talformatering:

Specifier	Type	Format	Output (Passed Double 1.42)	Output (Passed Int -12400)
c	Currency	{0:c}	$1.42	-$12,400
d	Decimal (Whole number)	{0:d}	System.FormatException	-12400

e	Scientific	{0:e}	1.420000e+000	-1.240000e+004
f	Fixed point	{0:f}	1.42	-12400.00
g	General	{0:g}	1.42	-12400
n	Number with commas for thousands	{0:n}	1.42	-12,400
r	Round trippable	{0:r}	1.42	System.FormatException
x	Hexadecimal	{0:x4}	System.FormatException	cf90

Specifik talformatering:

Specifier	Type	Example	Output (Passed Double 1500.42)	Note
0	Zero placeholder	{0:00.0000}	1500.4200	Pads with zeroes.
#	Digit placeholder	{0:(#).##}	(1500).42	
.	Decimal point	{0:0.0}	1500.4	
,	Thousand separator	{0:0,0}	1,500	Must be between two zeroes.
,.	Number scaling	{0:0,.}	2	Comma adjacent to Period scales by 1000.
%	Percent	{0:0%}	150042%	Multiplies by 100, adds % sign.
e	Exponent placeholder	{0:00e+0}	15e+2	Many exponent formats available.

4.19.6.10 *Int32*

(EB, 2015-08-05, Forms v1.1.0.344)

Validerar att ett värde är ett korrekt heltal. Denna gör i praktiken samma sak som stringInt32Converter. Användningsområdet dock vara kan vara att man inte vill typdefiniera ett fält som heltal men vill ändå specificera att fältet ska vara heltal.

Följande parametrar finns att ange för regeln:

Parameter	Beskrivning
[0]	Felmeddelande som ska visas vid valideringsfel. Ersätts med en standardtext från språkfilen om det är tomt eller utelämnas.

Följande parametrar går att använda i felmeddelandetexten:

Strängparameter	Beskrivning
{0}	Det angivna värdet i fältet.

4.19.6.10.1 Exempel

```
<TextBox.Tag>
  Heltal|xPath|stringEncodedConverter||
  |Mandatory;Int32;Angivet värde {0} är inte ett korrekt heltal.
</TextBox.Tag>
```

4.19.6.11 *Double*

(EB, 2015-08-05, Forms v1.1.0.344)

Validerar att angivet värde är ett korrekt flyttal. Denna gör i praktiken samma sak som stringDoubleConverter. Användningsområdet kan dock vara att man inte vill typdefiniera ett fält som flyttal men vill ändå specificera att fältet ska vara flyttal.

Följande parametrar finns att ange för regeln:

Parameter	Beskrivning
[0]	Felmeddelande som ska visas vid valideringsfel. Ersätts med en standardtext från språkfilen om det är tomt eller utelämnas.

Följande parametrar går att använda i felmeddelandetexten:

Strängparameter	Beskrivning
{0}	Det angivna värdet i fältet.

4.19.6.11.1 Exempel

```
<TextBox.Tag>
  Flyttal|xPath|stringEncodedConverter||
  |Mandatory;Double;Angivet värde {0} är inte ett korrekt flyttal.
</TextBox.Tag>
```

4.19.6.12 *StringTooLarge*

(EB, 2015-08-05, Forms v1.1.0.344)

Denna regel används på egen risk.

StringTooLarge är en validerings- och formateringsregel. Den är specifikt framtagen med tanke på summa fält för beräkningarna, där man inte kan veta i förväg hur stort tal man får tillbaks vid olika formelberäkningar. Vad den gör är att den kontrollera textfältets information och se om den får plats på textboxen eller inte. Om värdet inte får plats så resätts det med ”###”, notera också att de då kommer bli det värdet som sparas.

Följande parametrar finns att ange för regeln:

Parameter	Beskrivning
[0]	Kan ge textbox ett utseende som sin gömda stil. 0 = ingen ändring, 1 = använd alternativt utseende. Notera att det är mycket troligt att knappen Markera/Avmarkera fält i menyn troligen skriver över detta.
[1]	Felmeddelande som ska visas vid valideringsfel. Ersätts med en standardtext från språkfilen om det är tomt eller utelämnas.

Följande parametrar går att använda i felmeddelandetexten:

Strängparameter	Beskrivning
{0}	Det angivna värdet i fältet.

4.19.6.12.1 Exempel

```
<TextBox.Tag>
  Summa|xPath|stringEncodedConverter||
  |Mandatory;StringTooLarge;Angivet värde {0} är för stort för fältet.
</TextBox.Tag>
```

4.19.6.13 *GroupDependantMandatory*

(EB, 2015-08-05, Forms v1.1.0.344)

Validerar att ett fält har ett värde angivet. Ytterligare gemensamma inställningar för en grupp av fält måste göras i formsettings, även om inte vilka fält som ska ingå i gruppen behöver definieras. Till skillnad från GroupChoiceDependantMandatory kommer de olika fälten i samma grupp kommer inte alls att ha någon påverkan på varandra.

Regeln är tänkt att kopplas till att vara beroende på om ett annat fält har ett värde angivet eller inte. Skillnaden mot DependentMandatory är här att fältet kan sättas obligatoriskt om det beroende fältet inte har ett värde, annars kan DependentMandatory lika väl användas. **Referens:** Se kapitel 4.19.6.2 DependentMandatory.

Om inget beroende fält anges får regeln samma funktion som Mandatory. **Referens:** Se kapitel 4.19.6.1 Mandatory.

Notera att valideringsregeln tidigare hette ChbDepMandByPara, vilket fortfarande går att använda.

Följande parametrar finns att ange för regeln:

Parameter	Värden	Beskrivning
[0]	GroupName	Namnet på den grupp i Formsettings som innehåller de valideringsinställningar som ska användas. **Referens:** Se kapitel 4.19.6.13.1 Formsettings.

[1]	-/0/1	Bör inte längre användas. Styr vilken valideringsnivå som ska användas där 0 = Warning och 1 = Mandatory. Används istället huvudparametern för valideringsnivå. **Referens:** Se kapitel 4.19.1 Valideringsnivå.
[2]	Text	Felmeddelande som ska visas vid valideringsfel. Ersätts med en standardtext från språkfilen om det är tomt eller utelämnas.

Följande parametrar går att använda i felmeddelandetexten:

Strängparameter	Beskrivning
{0}	Namnet på det beroende fältet. **Referens:** Se kapitel 4.6.1 ElementName.
{1}	Text från språkfilen som säger om det beroende fältet är angett eller inte. Används som standard för att kunna få till ”Ett värde måste anges eftersom fältet Fyll i mig är inte angett”.

4.19.6.13.1 Formsettings

För att ange inställningar för gruppen behövs följande i formsettings:

```
<Validation active="1" debug="0">
  <Version>2.0</Version>
  <DependentMandatoryGroup name="GroupName">
    <IsDependent>1</IsDependent>
    <DependentXName>FieldName</DependentXName>
    <DependentParameter>1</DependentParameter>
  </DependentMandatoryGroup>
</Validation>
```

Parameter	Värden	Beskrivning
name	Text	Valfritt namn på gruppen som ska ange för regeln i fältets tag.
IsDependent	**1**/0	Om gruppen ska ha vara beroende av ett fält eller inte. Om elementet utelämnas kommer 1 att användas.
DependentXName	x:Name	Namnet på det beroende fältet som ska styra om fälten som använder denna grupp ska vara obligatoriska eller inte. Notera att ett korrekt fält måste anges och vara ett fält med en giltig tag och xPath. Kan utelämnas om `IsDependent` är satt till 0.
DependentParameter	1/0	1 = Fälten är obligatoriska om det beroende fältet har ett värde, 0 =

		Fälten är obligatoriska om det beroende fältet inte har ett värde. Om elementet utelämnas kommer 1 att användas. Kan utelämnas om `IsDependent` är satt till 0.

4.19.6.14 *GroupChoiceDependantMandatory*

(EB, 2015-08-05, Forms v1.1.0.344)

Validerar att ett definierat antal fält ur en grupp är ifyllda. Vilka fält som ingår i gruppen måste definieras i för gruppen gemensamma inställningar i formsettings. Notera att varje fält ändå måste ha regeln definierad i sin tag för att kunna visa valideringsmarkeringar.

Regeln är tänkt att kunna kopplas till att vara beroende på om ett annat fält har ett värde angivet eller inte. Annars valideras bara att rätt antal fält är ifyllda.

Notera att valideringsregeln tidigare hette ChbMxToMyDepMandByPara, vilket fortfarande går att använda.

Följande parametrar finns att ange för regeln:

Parameter	Värden	Beskrivning
[0]	GroupName	Namnet på den grupp i Formsettings som innehåller de valideringsinställningar som ska användas. **Referens:** Se kapitel 4.19.6.13.1 Formsettings.
[1]	-/0/1	Bör inte längre användas. Styr vilken valideringsnivå som ska användas där 0 = Warning och 1 = Mandatory. Används istället huvudparametern för valideringsnivå. **Referens:** Se kapitel 4.19.1 Valideringsnivå.
[2]	Text	Felmeddelande som ska visas vid valideringsfel. Ersätts med en standardtext från språkfilen om det är tomt eller utelämnas.

Följande parametrar går att använda i felmeddelandetexten:

Strängparameter	Beskrivning
{0}	Namnet på det beroende fältet. **Referens:** Se kapitel 4.6.1 ElementName.
{1}	Lägsta tillåtna antal ifyllda fält från gruppinställningarna. **Referens:** Se kapitel 4.19.6.14.1 Formsettings.
{2}	Högsta tillåtna antal ifyllda fält från gruppinställningarna. **Referens:** Se kapitel 4.19.6.14.1 Formsettings.

{3}	Text från språkfilen som säger om det beroende fältet är angett eller inte. Används som standard för att kunna få till ”Ett värde måste anges eftersom fältet Fyll i mig är inte angett”.

4.19.6.14.1 Formsettings

För att ange inställningar för gruppen behövs följande i formsettings:

```
<Validation active="1" debug="0">
  <Version>2.0</Version>
  <DependentMandatoryGroup name="GroupName">
    <IsDependent>1</IsDependent>
    <DependentXName>FieldName</DependentXName>
    <DependentParameter>1</DependentParameter>
    <SourceXName>
      FieldName1;FieldName2;FieldName3;FieldName4
    </SourceXName>
    <MinX>1</MinX>
    <MaxY>4</MaxY>
  </DependentMandatoryGroup>
</Validation>
```

Parameter	Värden	Beskrivning
name	Text	Valfritt namn på gruppen som ska ange för regeln i fältets tag.
IsDependent	**1**/0	Om gruppen ska ha vara beroende av ett fält eller inte. Om elementet utelämnas kommer 1 att användas.
DependentXName	x:Name	Namnet på det beroende fältet som ska styra om fälten som använder denna grupp ska vara obligatoriska eller inte. Notera att ett korrekt fält måste anges och vara ett fält med en giltig tag och xPath. Kan utelämnas om `IsDependent` är satt till 0.
DependentParameter	1/0	1 = Fälten är obligatoriska om det beroende fältet har ett värde, 0 = Fälten är obligatoriska om det beroende fältet inte har ett värde. Om elementet utelämnas kommer 1 att användas. Kan utelämnas om `IsDependent` är satt till 0.
SourceXName	x:Name	En lista med namnen på alla de fält som ska ingå i gruppen. Notera att varje fält ändå måste ha regeln för att kunna visa valideringsmarkeringar.

MinX	Heltal	Minsta antal fält i gruppen som måste vara ifyllda. Om inget eller ett felaktigt värde anges kommer Int32.MinValue att användas (alltså obegränsat).
MaxY	Heltal	Högsta antal fält i gruppen som får vara ifyllda. Om inget eller ett felaktigt värde anges kommer Int32.MaxValue att användas (alltså obegränsat).

4.19.6.15 *SystemValue*

(EB, 2015-08-05, Forms v1.1.0.344)

Validerar att ett fälts värde innehåller samma värde som en metadataparameter i systemet.

Följande parametrar finns att ange för regeln:

Parameter	Beskrivning
[0]	Definierat namn på det systemvärde som valideringen ska göras mot. **Referens:** Se kapitel 4.19.6.15.1 SystemValueType.
[1]	Felmeddelande som ska visas vid valideringsfel. Ersätts med en standardtext från språkfilen om det är tomt eller utelämnas.

Följande parametrar går att använda i felmeddelandetexten:

Strängparameter	Beskrivning
{0}	Det angivna värdet i fältet.
{1}	Angivet SystemValueType från parameter [0].

4.19.6.15.1 SystemValueType

SystemValueType	Beskrivning
Patient.Id	Patientens personnummer.
Patient.FullName	Patientens namn.

4.19.6.15.2 Exempel

```
<TextBox.Tag>
  Personnummer|xPath|stringEncodedConverter||
  |Mandatory;SystemValue;Patient.FullName:Patientens personnummer i blanketten är
felaktigt.
</TextBox.Tag>
```

4.19.6.16 *IdentificationNumber*

(EB, 2015-08-05, Forms v1.1.0.344)

Följande parametrar finns att ange för regeln:

Parameter	Beskrivning

[0]	Det definierade namnet på det nummerformat som valideringen ska göras mot. **Referens:** Se kapitel 4.19.6.16.1 IdNumberType.
[1]	Felmeddelande som ska visas vid valideringsfel. Ersätts med en standardtext från språkfilen om det är tomt eller utelämnas.
[2]	Lägesinställningar för regeln, kan användas för att omvända resultatet. **Referens:** Se kapitel 4.19.6.16.2 ValidationMode.

Följande parametrar går att använda i felmeddelandetexten:

Strängparameter	Beskrivning
{0}	Det angivna värdet i fältet.
{1}	Angivet IdNumberType från parameter [0].

4.19.6.16.1 IdNumberType

IdNumberType	Beskrivning
SE.Personnummer	Ett svenskt person- eller samordningsnummer.

4.19.6.16.2 ValidationMode

ValidationMode	Beskrivning
Invert	Valideringsregeln tolkas omvänt, alltså att värden av angivet format IdNumberType inte är godkända, men alla andra värden är det.

4.19.6.16.3 Exempel

```
<TextBox.Tag>
  Summa|xPath|stringEncodedConverter||
  |Mandatory;StringTooLarge;Angivet värde {0} är för stort för fältet.
</TextBox.Tag>
```

4.19.7 Förutsättning för att kunna visa valideringsmakeringar

För att textboxen ska kunna visa sina valideringsmarkingar `! ? *` så måste den vara direkt underelement till en Canvas. Anledning till det är att ED Forms måste kunna absolut positionera element med hjälp av Canvas elementet.

```
<Canvas>
        <TextBox x:Name="name1" Width="190" Style="{DynamicResource
        tbStyle}" Canvas.Top="1020" Canvas.Left="551">
             <TextBox.Tag>|xPath|stringEncodedConverter|</TextBox.Tag>
        </TextBox>
</Canvas>
```

4.19.8 Förutsättning för att felaktiga datatyper ska kunna sparas

I Blankett.xsd filen ska dessa tillägg läggas till, en complextype med namnet ErrorTextContainer och i complextype Blankett ska en underelement errortestlist tilläggas.

```
  <complexType name="ErrorTextContainer">
    <sequence>
      <element minOccurs="0" maxOccurs="unbounded" name="formgroup"
      type="tns:FormGroup" />
    </sequence>
    <attribute name="__Id" type="tns:guid" />
  </complexType>

  <complexType name="Blankett">
    <sequence>
      <element name="patient" type="tns:Patient" />
      <element name="sjukvenh" type="tns:Halsosjukvardsenhet" />
      <element name="sjukvpers" type="tns:Halsosjukvardspersonal" />
      <element name="ortdatum" type="tns:OrtDatum" />
      <element name="formlist" type="tns:FormList" />

      <element name="errorTextList" type="tns:ErrorTextContainer" />

   </sequence>
    <attribute name="__mainId" type="tns:guid" />
    <attribute name="__form" type="int" />
    <attribute name="__IBISVersion" type="int" />
    <attribute name="__status" type="int" />
    <attribute name="__version" type="dateTime" />
    <attribute name="__Id" type="tns:guid" />
  </complexType>
```

I Blanekett.xml filen ska dessa tilläggs göras:

```
…
  <errorTextList>
    <formgroup typ="text">
       <string4000value typ="ETC"><value/></string4000value>
    </formgroup>
  </errorTextList>
</blankett>
```

4.20 ClassificationCodes

(EB, 2014-12-17, Forms v1.1.0.335)

ClassificationCodes hanterar inställningar, hämtningar, sökning uthämtning och ifyllnad av koduppsättningar. Sedan läge har ICD10 funnits och varit hårdkodat in i Forms, men det går nu göra vissas inställningar för ICD10 i ClassificationCodes. Tanken är att denna ska kunna utökas mer fler funktioner och nya koduppsättningar för ifyllnadshjälp till användarna.

Notera att för tillfället kan enbart ICD10 används i vänsterfliken. Helper kan användas för flera sets och hanteras för samtliga sätt som är aktiverade, men stödjer å andra sidan enbart lokala koder och kan inte hämta koder från databas/tjänst än. Används av bland annat ICD10 (**Referens:** Se kapitel 4.1.3.2 ICD-10) och DataTransfer (**Referens:** Se kapitel 4.21 DataTransfer).

4.20.1 Formsettings

(EB, 2014-12-17, Forms v1.1.0.335)

```
<classificationcodes active="1">
  <classificationset type="ICD10" active="1">
    <subsets>
      <subset type="ICD-10-SE" label="Systematisk" active="1"
source="DataBase:ICD10SE"/>
      <subset type="KSH97-P" label="Primärvård" active="1" source="DataBase:KSH97P"/>
      <subset type="ESS" label="RETTS ESS" active="1" source="Local">
        <item code="R00" text="Onormal hjärtrytm" group="1"/>
        <item code="I48.9" text="Förmaksflimmer/fladder" group="1"/>
        <item code="R03.0" text="Högt blodtryck" group="2"/>
        <item code="R03.1 (som sökorsak)" text="Lågt blodtryck" group="2"/>
        <item code="R04.2" text="Blodhosta/Hemoptys" group="3"/>
        <item code="R04.0" text="Näsblödning" group="3"/>
      </subset>
    </subsets>
    <targets mode="Recent">
      <target active="1" type="Code" mode="Replace" format="{0}"
targetfield="Diagnoskod" label="Kod"/>
      <target active="1" type="Text" mode="Add" format=" {0}" targetfield="Diagnos"
label="Diagnos"/>
      <target active="1" type="CodeAndText" mode="Add" format=" {1} ({0})"
targetfield="Diagnos" label="Diag. och kod"/>
      <target active="1" visible="0" name="sokorsak1" type="CodeAndText" mode="Add"
format="{1} {0}" targetfield="sokorsak1" label="Sökorsak 1">
        <helper active="1" subset="ESS" allowcustomvalues="false"
allwaysdisplayvalues="true" itemlimit="200" displayformat="{1} {0}" />
      </target>
      <autotarget active="1" code="true" text="true" codeandtext="false" label="Auto"
/>
    </targets>
    <!--<settings>
            <setting
type="SearchableChars">A;B;C;D;E;F;G;H;I;J;K;L;M;N;O;P;Q;R;S;T;U;V;X;Y;Z;Å;Ä;Ö</settin
g>
            <setting type="TextSearchLabel">Diagnos</setting>
            <setting type="CodeSearchLabel">Kod</setting>
        </settings>-->
  </classificationset>
</classificationcodes>
```

Element	Möjliga värden	Beskrivning	Standardvärde (om inget anges)
classificationcodes/@active	1/0	Säger om koduppsättningar ska vara aktivt alls eller inte.	0, 1 för ICD10.
classificationset	-	Representerar en uppsättning av koder, finns för att kunna ha mer än en uppsättning.	ICD10 kommer "läggas till" om den inte finns med.

classificationset/@type	ICD10	Nyckel för kodsetet, för tillfället stödjs enbart ICD10.	-, felaktiga nycklar ignoreras.
classificationset/@active	1/0	Om kodsetet med tillhörande nyckel **@type** ska vara aktivt eller inte.	0, 1 för ICD10.
classificationset/subsets	-	Framtida möjlig konfiguration för att styra vilka undergrupperingar av koder som ingår i ett kodset.	-
subsets/@source	-	Framtida möjligt dynamiskt sätt för att styra var koderna ska hämtas ifrån, t.ex. då från databas med namnet på tabellerna som koderna finns i för att kunna lägga till dessa dynamiskt.	-
classificationset/targets/@mode	Recent/Target/Mixed	Styr hur knapparna för lägga in kod och text till fälten ska fungera. **Recent** = till senast aktivta fält, **Target** = till ett specificerat fält (se **@targetfield**), **Mixed** = användaren får själv välja mellan Recent och Target med ett val.	Recent
targets/target	-	Representerar var och en av knapparna för att flytta text/kod till fälten.	**Code**, **Text** och **CodeAndText** kommer alltid "läggas till" om de saknas.

targets/target/@active	1/0	Styr om knappen ska visas eller inte.	0, 1 för ICD10.
targets/target/@type	Code/Text/CodeAndText	Vilken av de tre knapparna som inställningarna gäller för. Enbart en target av varje typ ska finnas, överflödiga ignoreras.	-, Felaktiga värden ignoreras.
targets/target/@mode	Add/Replace	Hur text/kod ska läggas till fältet, **Add** = lägger till efter existerande text i fältet, **Replace** = raderar och ersätter existerande text. Är **targets/@mode = Recent** kommer **Add** alltid användas oberoende av vad som är valt.	Add
targets/target/@format	string	Hur text/kod ska formateras, använder sig av string.Format. För **target/@type = Code or Text** finns parameter {0}, för **CodeAndText** finns parametrar {0} och {1}. |string.Empty, Code och Text kommer formateras "{0}", CodeAndText "{0} {1}	

		, Code, Text"	
targets/target/@targetfield	string	x:Name på det fält i blanketten dit värdet ska fyllas på.	string.Empty, om namnet saknas eller är felaktigt kommer inget att hända om **targets/@mode = Target** eller **autotarget** används.
targets/target/@label	string	Den text som ska visas på knappen, kan lämnas tom för behålla texten från språkfilen.	string.Empty, värdet plockas från Lang.xml.
targets/autotarget	-	Knapp för att anropa de övriga knapparna automatiskt och alltid fylla i det/de specificerade fälten. Denna knapp kommer alltid hanteras som om **targets/@mode = Target** oberoende av konfiguration eller användarens val, vid **Recent** respektive **Mixed**.	En "tom" inaktiv skapas alltid, överflödiga ignoreras.
targets/autotarget/@active	1/0/true/false	Styr om knappen ska visas eller inte.	0
targets/autotarget/@code	1/0/true/false	Om knappen ska anropa **Code** med **targets/@mode = Target**.	0
targets/autotarget/@text	1/0/true/false	Om knappen ska anropa **Text** med **targets/@mode = Target**.	0
targets/autotarget/@codeandtext	1/0/true/false	Om knappen ska anropa **CodeAndText**	0

		med **targets/@mode = Target**.	
targets/autotarget/@label	string	Den text som ska visas på knappen, kan lämnas tom för behålla texten från språkfilen.	string.Empty, värdet plockas från Lang.xml.
targets/*/helper		En helper skapar. Kan används för samtliga target element, men ej för autotarget.	
settings	-	Ej implementerat	-

Om ICD10 inte finns med som classificationset kommer standardvärdena göra att nuvarande funktion för ICD10 används.
AutoTarget kommer dock vara inaktivt då detta kräver ett targetfield.
För att inte använda ICD10 måste någon av följande inställningar göras:
classificationcodes/@active = 0 eller classificationset(@type='ICD10')/@active = 0, och då kommer ICD10-fliken i Forms att inte visas.

När kod och/eller text läggs till fältet görs detta nu direkt in i ärende XML:et vilket betyder att till skillnad från tidigare fungerar detta oberoende av vilket/vilka fält som är kopplade till värdet och inte bara textboxar som tidigare.

Delen settings är inte implementerat utan enbart en tänkbar framtida fortsättning för att göra funktionen mer dynamisk och kunna stödja fler kodregister.

4.21 DataTransfer

(EB, 2015-08-05, Forms v1.1.0.344)

DataTransfer möjliggör förflyttning av värden mellan olika fält i blanketten och/eller platser i blankett-XML:et. Överföringar kan i dagsläget ske vid alla tillfällen då ett fält i blanketten, som är kopplat till blankett-XML:et, ändras av användaren. Det krävs alltså att fältet som ändras har en korrekt xPath.

Grundstrukturen för DataTransfer ser schematiskt ut enligt följande:

```
<datatransfer active="1">
  <transferset active="1">
    <trigger/>
    <transfer>
      <condition/>
      <sources>...</sources>
      <targets>...</targets>
    </transfer>
    <transfer>...</transfer>
```

```
    <transfer>...</transfer>
  </transferset>
  <transferset active="1">
    <trigger/>
    <trigger/>
    <transfer>
      <condition/>
      <condition/>
      <condition/>
      <sources>...</sources>
      <targets>...</targets>
    </transfer>
  </transferset>
  <transferset>...</transferset>
</datatransfer>
```

4.21.1Transferset

(EB, 2015-08-05, Forms v1.1.0.344)

Under `datatransfer`, kan man sedan ha ett antal `transferset`. Varje `transferset` är ett set av gemensamma överföringar. Ett `transferset` delar samma uppsättning av `trigger` och alla överföringar i ett sätt kommer därför alltid ske vid samma tillfälle om dess respektive `condition` är uppfyllda.

4.21.2 Trigger

(EB, 2015-08-05, Forms v1.1.0.344)

För att styra när överföringar ska ske används en eller flera `trigger`. Genom att sätta `required="true"` på samtliga `trigger` kan måste alla vara uppfyllda för att överföring ska ske. Anges det inte som kommer överföring att ske när någon (minst 1st) av dem är uppfylld(a).

Notera att om `required="true"` ska användas bör samtliga `trigger` göra det, annars kommer de som inte har det/inte har det aktivt att inte fylla någon funktion. Finns det bara en `trigger` så kommer `required` inte att fylla någon funktion alls då det bara finns en och den måste vara uppfylld.

En `trigger` ser ut på följande sätt:

```
<trigger triggername="" triggertype="" triggervalue="" required="" inverted=""/>
```

Parameter	Värden	Beskrivning
triggername	x:Name	Namnet på det fält i blanketten som ska trigga överföringen.
triggertype	*	Vad på fältet som ska avgöra om det triggar överföringen vid ett specifikt tillfälle. **Referens:** Se kapitel 4.21.2.1 TriggerType för godkända värden.
triggervalue	*	Vilket värde som ska trigga överföringen. Vilka värden som är godkända beror på vilken `triggertype`

		som är angiven. **Referens:** Se kapitel 4.21.2.1 TriggerType för godkända värden.
required	true/**false**	Om denna `trigger` är obligatorisk eller inte. Denna behöver inte anges.
inverted	true/**false**	Om resultatet av kontrollen ska inverteras för att ge motsatt resultat. Denna behöver inte anges.

4.21.2.1 *TriggerType*

(EB, 2015-08-05, Forms v1.1.0.344)

Följande är de olika värden som kan användas för `triggertype` som styr vad på fältet som ska trigga överföringen:

TriggerType	Beskrivning
HasAValue	Att fältet har ett värde angivet och inte är tomt. Godkända värde för `triggervalue` är: true/false.
Value	Att fältets värde är lika med angivet `triggervalue`.
ValueChanged	Att fältets värde ändras, oavsett vad det ändras till. Godkända värde för `triggervalue` är: true/false.
ValueContains	Att fältets värde innehåller angivet `triggervalue`.
Property	Att ett attribut på fältet har ett specifikt värde. Fås fram med reflection. Godkända värde för `triggervalue` är: {0}:{1} där {0} = attributets namn och {1} = attributets värde, t.ex. Text:Röntgen. Kontrollen mot attributets värde görs med object.ToString(), så är värdet ett objekt kommer jämförelsen sannolikt göras med objektets namn, t.ex. för attributet Style.

4.21.2.2 *Exempel*

(EB, 2015-08-05, Forms v1.1.0.344)

Följande överföring görs om fältet med namnet `FieldOne` inte har ett värde angivet:

```
<trigger triggername="FieldOne" triggertype="HasAValue" triggervalue="false" />
```

4.21.3 Transfer

(EB, 2015-08-05, Forms v1.1.0.344)

En `transfer` representerar en överföring av ett och samma värde till ett eller flera fält. Värdet som ska överföras kan dock slås samman från olika källor som definieras i en `sources` nod. Fältet eller fälten som överföringen ska göras till definieras i en `targets` nod. Det kan idag enbart finnas en `sources` och en `targets` nod.

Men bara för att den överliggande `transferset` föräldern som en `transfer` tillhör triggar en överföring betyder det inte säkert att överföringen blir av. Detta då en eller flera `condition` noder kan definieras upp för att styra om överföringen ska göras. **Referens:** Se kapitel 4.21.4 Condition.

Följande är en schematisk bild över hur en `transfer` ser ut:

```
<transfer>
  <condition/>
  <condition/>
  <condition/>
  <sources>...</sources>
  <targets>...</targets>
</transfer>
```

4.21.4 Condition

(EB, 2015-08-05, Forms v1.1.0.344)

En `condition` används för att utöka kontrollerna för ifall en specifik `transfer` ska utföra sin överföring eller inte efter att de gemensamma villkoren har uppfyllts. Om ingen extra kontroll behöver göras utan överföringen ska ske oavsett så används ingen `condition` utan utelämnas helt.

En `condition` är uppbyggd på exakt samma sätt som en `trigger`, vilket även gäller att en eller flera kan användas och att samtliga bör ha samma inställning för `required` för att få dem att kräva alla eller någon (minst 1st) av alla `condition` uppfyllda för att tillhörande `transfer` ska genomföra sin överföring. **Referens:** Se kapitel 4.21.2 Trigger.

```
<condition triggername="" triggertype="" triggervalue="" required="" inverted=""/>
```

4.21.5 Sources

(EB, 2015-08-05, Forms v1.1.0.344)

En `sources` används för att styra vilka värden som ska överföras. Samtliga värden som anges kommer hämtas ut, eventuellt rensas och trimmas om `ignoreempty` respektive `trim` är aktivt och slås samman till ett värde med angivet format enligt `format` innan det överförs till angivna mål i `targets`.

```
<sources sourcetype="" ignoreempty="" trim="" format="">source1;source2</sources>
```

Parameter	Värden	Beskrivning
sourcetype	*	Styr vilken typ av värden som ska hämtas. **Referens:** Se kapitel 4.21.7 TransferElementType.
ignoreempty	true/**false**	Om tomma värden ska exkluderas i överföringen.

trim	**true**/false	Om värden ska trimmas innan de överförs. Notera att detta enbart gäller värdets innehåll och inte definitionen av vilket värde som ska användas då det alltid trimmas förutom om `sourcetype` är text då definitionen och värdet är samma sak.
format	*	Vilket format som ska användas vid överföringen av värdena. **Referens:** Se kapitel 4.21.5.1 Format.
Sources.value	*	List över vilka värden som ska hämtas, varje värde separeras med semikolon (;). Vad som ska anges beror på vilken `sourcetype` som är angiven. **Referens:** Se kapitel 4.21.7 TransferElementType för hur dessa ska anges.

4.21.5.1 *Format*

(EB, 2015-08-05, Forms v1.1.0.344)

Format anger hur värdena ska formateras i förhållande till varandra när de slås samman för att överföras. Detta görs på tre olika sätt beroende på hur `format` är angivet.

4.21.5.1.1 Tomt format

(EB, 2015-08-05, Forms v1.1.0.344)

Om inget `format` anges/lämnas tomt kommer värdena slås samman rakt av enligt: ”A”, ”B”, ”C” ➔ ”ABC”. Om det bara är ett värde som hämtas och det ska hämtas rakt av behövs inget `format` anges.

4.21.5.1.2 String.Format

(EB, 2015-08-05, Forms v1.1.0.344)

Som andra alternativ kan normal format enligt `string.Format` användas. Det är dock viktigt att detta anges korrekt annars kommer överföringen misslyckas. Det är också starkt rekommenderat att inte aktivera `ignoreempty` då det kan betyda att värden och parametrar saknas till `string.Format` vilket också kommer innebära att överföringen misslyckas.

4.21.5.1.3 Specialformat

(EB, 2015-08-05, Forms v1.1.0.344)

Det tredje alternativet är att ange ett specialformat där man kan lägga till text eller separerare mellan varje värde som inkluderas med eller utan utrensning av tomma värden enligt följande:

{o}{s}{1}{*}{2}{n}{3}
Där parametrarna betyder följande:

Parameter	Beskrivning
{0}	Text före första värdet. Ersätts med den text som ska läggas till t.ex. startparentes ”(”.
{s}	Första värdet. Anges exakt som ”{s}”.
{1}	Text före varje värde, inklusive första och sista. Ersätts med den text som ska läggas till t.ex. bindestreck ”-”.
{*}	Varje värde, inklusive första och sista. Anges exakt som ”{*}”.
{2}	Text efter varje värde, inklusive första och sista. Ersätts med den text som ska läggas till t.ex. kommatecken och mellanslag ”, ”.
{n}	Sista värdet. Anges exakt som ”{n}”.
{3}	Text efter sista värdet. Ersätts med den text som ska läggas till t.ex. slutparentes ”)”.

För tre värden ”A”, ”B”, ”C” skulle detta kunna användas på följande sätt:

- Definition som ska användas i `format`: ”({s}-{*}, {n})”.
- Logiskt ses som: {0}{1}A{2}{1}B{2}{1}C{2}{3} där {0} = ”(”, {1} = ”-”, {2} = ”, ”, {3} = ”)” .
- Ge det slutliga resultatet att överföra: ”(-A, -B, -C,)”.

Så för att få till en extra text/avskiljare före första värdet används alltså ”({s}” för en parentes (() och för att göra motsvarande efter sista värdet används ”{n})”. Vill man inte ha dessa parenteser kan man utesluta {s} och {n} och istället använda ”{*}, ”

För att exkludera specifika separerare finns speciella nyckelvärden som kan användas:

Nyckelvärde	Beskrivning
[NoStartPart]	Exkluderar separerare före första värdet. Måste komma först av allt för att fungera.
[NoOpeningDelimiter]	Exkluderar separerare före nästkommande värde. Det får inte komma något annat mellan denna och det nästkommande värdet för att den ska fungera.
[NoClosingDelimiter]	Exkluderar separerare efter föregående värde. Det får inte komma något annat mellan denna och det föregående värdet för att den ska fungera.
[NoEndPart]	Exkluderar separerare efter sista värdet. Måste komma allra sist för att fungera.
[NewLine]	Lägger till en ny rad.
[Space]	Lägger till ett mellanslag.

För att de nyckelvärden som ska exkludera separerare ska fungera måste de komma på den plats där separeraren skulle komma, annars ignoreras den. De kan under den förutsättningen dock finnas både som en del av ett annat värde eller som ett eget värde mellan två andra.

Övriga nyckelvärden som inte exkluderar separatorer kan placeras var som i ordningen så länge de inte är i vägen och stör ut något annat nyckelvärde.

OK:
”[NoOpeningDelimiter]Värde1;[NewLine]Värde2[Space]”
”[NoOpeningDelimiter]Värde1;[NoOpeningDelimiter];Värde2”
”[NoOpeningDelimiter];Värde1[NoOpeningDelimiter];Värde2” – Även om Värde2s parameter tillhör Värde1 så kommer det ändå fungera eftersom det inte kommer något annat emellan.

Fel:
”Värde1;[NoOpeningDelimiter];[NoClosingDelimiter];Värde2” – Eftersom båda sitter på fel plats kommer de inte ha någon effekt.

Notera om `XPath` eller `FieldName` används som `sourcetype` så måste dessa nyckelvärden finnas antingen som en del av värdet, vilket inte är så troligt, eller måste det finnas ett fält förifyllt i blankett-XML:et med dessa nyckelvärden. Detta fält kan då återanvändas på alla de ställen det behövs.

Så igen för nu istället fem värden ”A”, ”[NoOpeningDelimiter]”, ”B”, ”C”, ”[NoClosingDelimiter]” skulle detta kunna användas på följande sätt:

- Definition som ska användas i `format`: ”({s}-{*}, {n})”.
- Logiskt ses som: {0}{1}A{2}B{2}{1}C{3} där {0} = ”(”, {1} = ”-”, {2} = ”, ”, {3} = ”)” .
- Ge det slutliga resultatet att överföra: ”(-A, B, -C)”.

4.21.6 Targets

(EB, 2015-08-05, Forms v1.1.0.344)

Till sist används sedan `targets` för att ange de mål dit värdet ska överföras. Det är samma värde som kommer överföras till samtliga mål som sätts upp. Värdena alltid överföras direkt in i blankett-XML:et då även om fält eller grupper av fält kan anges som mål så används de för att hämta ut rätt xPath att överföra värdet till.

```
<targets targettype="" mode="">target1</targets>
```

Parameter	Värden	Beskrivning
targettype	*	Styr var vilken typ av mål som värdena ska överföras. **Referens:** Se kapitel 4.21.7 TransferElementType.
mode	*	Styr hur värdet ska överföra till målet.

Targets.value	*	List över vart värdet ska överföras, varje mål separeras med semikolon (;). Vad som ska anges beror på vilken `targettype` som är angiven. **Referens:** Se kapitel 4.21.7 TransferElementType för hur dessa ska anges.

4.21.6.1 *Mode*

(EB, 2015-08-05, Forms v1.1.0.344)

För att styra hur värdet ska överföras till målet används `mode`:

Mode	Beskrivning
Add	Lägger till värdet att föra över till målets existerande värde.
Replace	Ersätter målets tidigare värde med det nya överförda värdet.

4.21.7 TransferElementType

(EB, 2015-08-05, Forms v1.1.0.344)

Används för att styra hur värden ska hämtas med `sourcetype`, och vart de ska överföras till med `targettype`.

TransferElementType	Beskrivning
XPath	Värden ska hämtas från/överföras till xPath i blankett-XML:et. Varje xPath ska separeras med semikolon (;) om flera ska användas.
Children	Värden ska hämtas från/överföras till samtliga underelement till angivna element som är fält i blanketten med en giltig xPath. Fälten som är förälder ska anges med x:Name och separeras med semikolon (;) om flera ska användas.
FieldName	Värden ska hämtas från/överföras till angivna fält. Varje fält ska anges med x:Name och separeras med semikolon (;) om flera ska användas. Notera att fältet måste ha en giltig xPath för att komma med.
Text	Värden ska hämtas som rå text. Angivna värden hämtas rakt av och separeras med semikolon (;) om flera ska användas. Kan inte användas som mål för överföring.

4.22 Övriga funktioner

Beskrivning av diverse övriga funktioner i EyeDoc Forms.

4.22.1 Kortkommandon

Namn	Beskrivning
Ctrl end	Sist i blanketten eller sist i aktuellt fält
Ctrl home	Först i blanketten eller först i aktuellt fält
Ctrl G	Förlegad: Spara till Signeringskorg
Ctrl S	Förlegad: Spara till Skrivkorg
Ctrl D	Förlegad: Delsignera
Ctrl C	Kopiera
Ctrl X	Klipp ut
Ctrl V	Klistra in
Ctrl P	Förlegad: utskrift

Tabell: Kortkommandon.

4.22.2 Ritfunktion

(EB, 2012-08-01, Forms v1.1.0.275)

4.22.2.1 *Formsettings*

För att aktivera ritfunktionen i blanketten ska följande läggas till i blankettens `formsettings`:

```
<formsettings settingversion="1" formversion="1">
    <Paint active="1">
        <Version>1.0</Version>
    </Paint>
</formsettings>
```

4.22.2.2 *InkCanvas*

Själva ytan som där det ska gå att rita är en InkCanvas som sätts upp som ett helt vanligt formulärfält. Den datatyp dit dess xPath kopplas ska vara ett obegränsat strängvärde då den vektoriserade data kan bli relativt stor. Det går utmärkt att använda flera InkCanvas i en och samma blankett.

```
<InkCanvas x:Name="Cold_Where" Canvas.Top="645" Canvas.Left="400" Width="355" Height="190">

<InkCanvas.Tag>|formlist/formgroup[@typ='diagnos']/stringUnboundValue[@typ='cold_where']/value|stringEn
codedConverter|</InkCanvas.Tag>
</InkCanvas>

<InkCanvas x:Name="Signature" Width="343" Height="120" Canvas.Top="865" Canvas.Left="408">

<InkCanvas.Tag>|formlist/formgroup[@typ='signature']/stringUnboundValue[@typ='signature']/value|stringE
ncodedConverter|</InkCanvas.Tag>
</InkCanvas>
```

4.23 Övriga blankettfiler

4.23.1 Blankettschema (.xsd)

(EB, 2011-02-21, Forms v1.1.0.178)

XSD-filer ska inte längre göras utifrån varje blankett (som det gjorts tidigare), utan en av de aktiva mallarna ska alltid användas. Dessa får endast modifieras genom att nya enumeration lägg till.

Nya komplexa typer bör ej skapas för varje blankett. Till så stor utsträckning som möjligt ska de olika typerna som är fördefinierade .
MÅSTE VARA I RÄTT ORDNING
Alla element måste komma i rätt ordning oberoende om de är simpla eller komplexa, detta både för att det ska vara säkert att det fungerar och att det ska gå igenom IHT med full prestanda. Prestandamässigt är det dock fördelaktigt att komplexa typer som inte används i XML:et plockas inte användas , det är då viktigt att tänka på att om man blockar bort något och sedan behöver det igen så måste de placeras rätt, lättast är då att alltid kopiera från Schemamallen. Simpla typer får ALDRIG plockas bort ur en komplex typ.

Enum:
Enumeration ska används vid alla tillfällen som en komplex typ kan komma att användas mer

För att underlätta validering av elementen i schemat så bör minOccurs och maxOccurs användas och fungerar praktiskt alltid så att:
`minOccurs="1" maxOccurs="1"` = Simpel typ
`minOccurs="0" maxOccurs="unbounded"` = Komplex typ
a

`minOccurs="0" maxOccurs="1"` = Komplex typ som inte har någon enum, eller som av annan anledning inte ska få förekomma mer än 1 ggr. (Det finns ingen begränsning i Innehållstjänsten som fungerar på detta sätt utan är endast endast till för blankettproducentens skull)

4.23.2 Blankettdata (.xml)

(EB, 2011-02-21, Forms v1.1.0.178)

Har den komplexa typen enum så ska alla dessa enum alltid anges i XML och xPath oberoende om bara en kommer att användas, finns flera enum så ska alla anges. Detta är i första hand för att underlätta uppdateringar av blanketten men kan också ha statistiska syften då enumerationerna sparas ner på samma rad i databasen som värdet.

Namespace:et i XML:et används inte av innehållstjänsten vid sparning men däremot av Forms, vilket betyder att om det saknas så kommer inte blanketten att öppnas. Detta namespace måste också stämma överrens med det som finns angivet i schemat som ska sluta med blankettens artikelnummer enligt `http://gainit.se/IBIS/BLANKETT/<artnr>`.

4.23.3 Förifyllnad (MAP_EBIM.xml)

(EB, 2011-02-21, Forms v1.1.0.178)

Eller MEP_EBIM är definitionen för vilka värden som ska förifyllas I blanketten och är uppbyggt med en `row` per värden som ska förifyllas. Varje `row` består alltid av en källa (`source`) och ett mål (`target`) efter följande format:

```
<row>
  <source>
    <item type="xpathformat">BOOKING_DATE</item>
    <item type="text" action="format" argument="date">yyyy-MM-dd</item>
  </source>
  <target>
    <item action="create"
type="xpath">formlist/formgroup[@typ='grupp']/string100value[@typ='date']/v
alue</item>
  </target>
</row>
<row>
  <source>
    <item type="xpathformat">BOOKING_DATE</item>
    <item type="text" action="format" argument="date">HH:mm</item>
  </source>
  <target>
    <item action="create"
type="xpath">formlist/formgroup[@typ='grupp']/string50value[@typ='time']/va
lue</item>
  </target>
</row>
```

4.23.3.1 *Type*

`type` styr vad för typ av värde
`xpathformat` är förifyllnads-variabler som kommer från EyeDoc Connector, `xpath` är en xPath i blanketten och `text` är.
För `target` måste alltid `xpath` eftersom värdet alltid ska hämtas

4.23.3.2 *Action*

`format` = formatering av källvärdet, fungerar endast på `source`. Till `format` används också `argument` som tillägg.
`replace` /`create` = tar bort och ersätter eventuellt existerande värde eller skapar ett nytt värde. Båda dessa kommer att skriva över värden som är definierade både i XML-filen för mål-xPath:en och tidigare `item` i `source` på samma `row`.
`add` = lägger på värdet för det aktuella värdet till det som redan finns på målet. Vill man lägga ihop flera värden måste `add` användas istället på efterföljande `item` och vill man lägga ihop värdet med ett värde från XML-filen så måste alla använda `add`. Fungerar endast på `source`, vill man ha värden till flera xPath måste man använda sig av flera `row`.

4.23.4 Kopiefält (MAP_COPY.xml)

(EB, 2011-02-21, Forms v1.1.0.178)

Kopiefält i COPY filen fungerar i princip på samma sätt som EBIM. Skillnaden är att i den globala delen så `target` och `source` exakt lika för att kopiera information från en blankett till en ny version av samma blankett. Dessutom under varje row så ska både source och target vara en xPath i blanketten. Normalt är det samma xPath i båda för att värdet från den gamla versionen ska hamna i samma fält i den nya versionen. Men teoretiskt sett kan man lika väl kopiera från ett fält i den gamla till ett helt annat fält i den nya.

Viktigt att tänka på är att en och samma xPath inte kan användas som `target` i både EBIM och COPY eftersom ett förifyllt värde alltid skriver över ett kopievärde. Därför kommer i praktiken kopievärden aldrig komma in i den nya blanketten, det hjälper inte även om varken värde eller namn skickas från anropande system.

4.23.5 Fältinformation (GUI.xml)

(EB, 2014-12-19, Forms v1.1.0.335)

Överför blankettdata till gränssnitt. Fungerar på liknande sätt som EBIM/MAP_COPY.

4.23.6 Returdata (RetTra.xml)

(EB, 2014-12-19, Forms v1.1.0.335)

Överför blankettdata till andra system, t.ex. Patientliggare, Journalsystem. Fungerar på liknande sätt som EBIM/MAP_COPY.

4.23.7 DigitalPen tolkningar (EDPen.xml)

(EB, 2014-12-19, Forms v1.1.0.335)

Hanterar koppling och överföring av tolkningar till blankettdata. Fungerar på liknande sätt som EBIM/MAP_COPY.

4.24 Funktioner i XAML i Forms

Då Forms är baserat på en XAML blankett så går större delen av de inbyggda funktionerna och attributen på element i XAML. Den specifika implementationen av dessa XAML blanketter i Forms gör dock att vissa funktioner inte fungerar rakt av. Utöver detta finns flera helt egna funktioner på element vilka beskrivits i andra delar av detta dokument.
Detta stycke beskriver både några av dessa vanliga funktioner som finns i för specifika element XAML och hur dessa fungerar i Forms.

4.24.1 Grunduppbyggnad

Alla blanketter i Forms måste följa samma grunduppbyggnad. Något som ska noteras här är att alla sidor (`FixedPage`) alltid ska namnges `Page1`, `Page2`, `Page3`, osv.

```
<StackPanel x:Name="form1"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
  <StackPanel.Resources>
    <ResourceDictionary>
      <!--Stilmallar-->
      <!--Träd-->
      <!--Blankettinställningar-->
    </ResourceDictionary>
  </StackPanel.Resources>
  <Border Padding="0" Margin="0" BorderBrush="Gray" BorderThickness="1"
Width="793.76" Height="1122.56">
    <FixedPage Name="Page1" Width="793.76" Height="1122.56">
```

```
      <!--Layout-->
      <!--Ifyllnadsfält-->
    </FixedPage>
    <FixedPage Name="Page2" Width="793.76" Height="1122.56">
      <!--Layout-->
      <!--Ifyllnadsfält-->
    </FixedPage>
  </Border>
</StackPanel>
```

En blankett utan fasta sidor fungerar på exakt samma sätt som en vanlig blankett med fasta sidor men istället för `Border/FixedPage` enligt ovan så används en `StackPanel` utan någon höjd. Inuti denna placeras sedan allt innehåll i blanketten.

```
<StackPanel Width="793.76">
  <!--Layout-->
  <!--Ifyllnadsfält-->
</StackPanel>
```

Det går också att blanda fasta sidor med flytande innehåll fritt, dock går det ändå inte skriva ut de fasta sidorna om blanketten innehåller något annat än fasta sidor. Det går även att ha informationssidor i en blankett med flytande innehåll, dessa kommer i skillnad till vanliga sidor gå att skriva ut eftersom det under informationsfliken enbart kan finnas fasta sidor.

Alla blanketter i Forms har bredden `793.76` enligt ovan, detta motsvarar bredden på en A4 sida och för att det ska se bra ut ska detta värde alltid användas. Form är låst vid att visa denna bredd så om någon annan bredd anges kommer innehållet antingen inte fylla upp hela programfönstret eller hamna utanför och inte synas.

4.24.2 Informationssidor

För att en sida ska visas som en informationssida, alltså I den separata fliken ”Information” i Forms vid sidan av ”Formulär” så ska följande `Tag` attribut läggas till för den aktuella sidan (`FixedPage`).

```
<FixedPage.Tag>Target=2|PrintMode=1|||</FixedPage.Tag>
```

I dagsläget görs ingen kontroll på vad `FixedPage.Tag` innehåller utan enbart om den finns eller inte. `Target` och `PrintMode` tar alltså ingen hänsyn till vilket värde de sätts till, men då detta ev. kan komma att ändras i framtiden ska den sättas enligt ovan tillsvidare.
Dock kan enbart en `FixedPage` användas som informationssida och den kommer enbart accepteras som infosida om den ligger direkt under den yttersta `StackPanel`:n.

Det går att placera ifyllnadsfält på samma sätt som på en vanlig sida, däremot bör dessa kanske vara låsta. Detta kan användas för att få med ifylld data från blanketten på infosidorna vid visning och utskrift.

4.24.3 Textfält

Pseudokod:
Om (text på alla rader)

- Om (tangent tryckning ger ny rad) - ta bort det tecknet.
- Om (inklistrad text ger ny rad) - ta bort, från slutet av det inklistrade så det får plats.
- Om (tangent är mellanslag) - kontrollera att den får plats i textrutan.

4.24.4 Layout

4.24.4.1 *Glyphs*

Glyphs är fast text. Glyphs är det format som man får på text som standard från en XPS. Istället för Glyphs kan man också använda ett TextBlock utan ett `Tag` attribut för att få in fast text i blanketten (finns `Tag` så kommer Forms alltid att skriva över texten i objektet).

`UnicodeString` innehåller texten i Unicode. För reserverade XML-tecken måste `&` för & (och), `<` för < (mindre än), `>` för > (större än), `&quot;` för " (citat) samt `&apos;` för ' (apostrof) användas.
`FontUri` är den fysiska adressen till den typsnittsfil som ska användas och måste gå mot de fonter som finns i resources mappen genom t.ex `/resources/Fonts/timesbd.ttf`. Detta då `FontUri` stöder inte referenser till systemtypsnitt.
`FontRenderingEmSize` är textstorleken (se kapitel Fonter för storlekstabell)
`Fill` är textfärgen i alpha-RGB eller alternativt dess färgnamn (t.ex. `#ff000000` eller `Black`).
`StyleSimulations` kan användas för att simulera kursiv och fet stil, finns just det teckensnittet i Forms bör det användas istället då `StyleSimulations` inte ger exakt samma resultat.
`Indices` är en kodning av tecken för tecken tillsammans med mellanrum och ska inte användas i Forms blanketter utan plockas bort. Dels för att det är svårt att hantera i blanketten vid ändringar m.m., men också för att det kan skapa skillnader i utseenden mellan blanketten i Forms och en eventuell export till annat system/format/funktion.
`OriginX` & `OriginY` styr som standard dess position. Glyphs kan dock även positioneras på samma sätt som övriga XAML objekt.

4.24.4.2 *Path*

Path är vektoriserad layout vilka formar olika vektoriserade grafiska objekt och är svåra att skapa och redigera manuellt. En Path kan innehålla ett obegränsat antal vektordefinitioner i sitt `Data` attribut och innehåller också annan information som fyllfärg, linjetyp, tjocklek med mera.

Har man vektoriserade bilder och annan layout i sitt original innan det konverteras till XPS så kommer dessa att komma in i XAML som Path. Man kan även konvertera vektoriserade bilder till SVG, skriva ut dem till XPS och importera dem i blanketten.

4.24.4.3 *Rectangle & line*

För att skapa enklare grafiska element kan man använda sig av Rectangle och Line.
`Width` & `Height` styr dess storlekar
`Stroke` styr vilken färg de ska ha
`StrokeThickness` styr tjockleken på linjerna.
Vill man ha en ram runt ett objekt, t.ex. en TextBox eller en Grid cell, så är det bättre att använda Border istället. Detta för att man kan placera objektet direkt i Border och automatiskt anpassa sig till vad som ligger i medans med Rectangle skulle man bli tvungen att positionera ramen och objektet separat.

4.24.4.4 *Expanderbara fält*

(EB, 2011-02-21, Forms v1.1.0.178)

Expanderbara fält är något som främst används i blanketter utan fasta sidor. De går att använda på fasta sidor men där är det begränsad nytt med detta eftersom det inte går expandera från en sida till en annan och

4.24.4.5 *WrapPanel*

En `WrapPanel` är en behållare för andra objekt som alla kommer att placeras efter varandra horisontellt. Objekt i en `WrapPanel` kommer normalt att placeras till höger/vänster om varandra, men om hela dess bredd är fylld så kommer objekten att fortsätta på nästa ”rad”. Finns ingen plats för nya ”rader” beroende på vilken höjd som är satt så kommer objekten inte att synas. Med `VerticalAlignment` kan man styra om objekt ska placeras till i över- eller underkanten men detta styrs inte av höjden på wrappaneln utan på höjden av ”raden” som fås av det högsta objektet på densamma.

4.24.4.6 *StackPanel*

En `StackPanel` är en behållare för andra objekt som alla kommer att placeras efter varandra vertikalt. Med `HorizontalAlignment` kan man styra om objekt ska placeras till höger eller vänster. Objekt i en `StackPanel` kommer alltid att placeras under/över varandra, finns inte tillräcklig plats beroende på vilken höjd som är satt på `StackPanel` så kommer överflödiga element inte att synas.

4.24.4.7 *Expander*

En `Expander` kan användas för att ge användaren möjlighet att minimera och expandera delar i blanketten för att göra det smidigare att fylla i och navigera i denna. Detta särskilt i Standardvårdplaner då dessa ofta kan vara långa och innehålla mycket fält.

```
<Expander ExpandDirection="Down" IsExpanded="True">
  <Expander.Header>
   <TextBlock>Rubrik</TextBlock>
  </Expander.Header>
  <StackPanel>
```

```
    <!—Layout och Ifyllnadsfält -->
  </StackPanel>
</Expander>
```

Om en `Expander` är minimerad eller expanderad styr användaren själv, på varje expander kan man sätta vilket av dessa lägen om ska användas som standard men hjälp av `IsExpanded`. Sätts den till `True` kommer den alltid vara expanderad när man öppnar blanketten, sätts den till `False` kommer den alltid vara minimerad från början.

En Expander ger användaren möjlighet att expandera/minimera innehåll i blanketten. `ExpandDirection` styr åt vilket håll innehållet expanderar och är normalt `="Down"`. `IsExpanded` styr om Expandern som standard är expanderad eller minimerad med `True/False`.
Att använda Expander är framförallt användbart i blanketter med flytande uppbyggnad utan FixedPage för att göra det lättare för användaren att hitta i större blanketter. Detta genom att användaren kan dölja det som inte är intressant och bara visa den del av blanketten som denne ska arbeta med för tillfället.

```
<Expander ExpandDirection="Down" IsExpanded="False">
    <Expander.Header>
        Rubrik 1
    </Expander.Header>
```

En Expander kan bara innehålla ett element, övriga objekt får sedan placeras i detta. Då ExpandDirection normalt är Down så bör detta element vara en StackPanel.

```
    <StackPanel>
        Innehåll
    </StackPanel>
</StackPanel>
```

Notis: Pga. en bugg i Forms kan en Expander inte innehålla en Border som första element.

4.24.4.8 *Grid*

En Grid fungerar som en tabell. Varje kolumn och rad måste definieras upp på följande sätt:

```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="85"/>
        <ColumnDefinition Width="31"/>
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
        <RowDefinition Height="18"/>
        <RowDefinition Height="20"/>
    </Grid.RowDefinitions>
```

Detta skapar en 2x2 tabell, alla objekt ska sedan placeras innan dess sluttag och vilken cell de ska finnas i definieras enligt endan. Vill man ha flera objekt i samma cell vill man oftast lägga dessa i ett omslutande element eftersom cellerna inte flyter objekt i dem utan de kan komma att hamna Denna definition av cell behövs endast på det yttersta objektet så om flera objekt ska finnas i samma cell kan man lägga dem i t.ex. en Stack eller WrapPanel och ha denna

definition på dem istället för varje objekt inuti. Anges vilken kolumn och rad objektet ska ha så hamnar det i den första (0:0).

```
    <TextBlock Grid.Column="0" Grid.Row="0" Text="1:1"/>
    <TextBlock Grid.Column="0" Grid.Row="1" Text="1:2"/>
    <Border Grid.Column="1" Grid.Row="0">
        <TextBlock Text="2:1"/>
    </Border>
    <TextBlock Grid.Column="1" Grid.Row="1" Text="2:2"/>
</Grid>
```

Objekt i en cell kommer som standard placeras mitt i cellen. Detta går styra med attributen `HorizontalAlignment` och `VerticalAlignment`. Har objekten ingen storlek satt så kommer de däremot att ärva storleken från cellens definition och kommer därför fylla upp hela cellens bredd respektive höjd.
För att skapa mellanrum mellan objekt i Grid så kan `Margin` användas.

4.24.4.9 *Border*

Border kan användas för att skapa ramar runt objekt och bakgrunder till dem. Border tillför ingen annan funktion mer än detta och att den ev. används av Forms när det gäller blanketter med FixedPage.

4.24.4.10 *Canvas*

Canvas fungerar som en behållare för andra objekt. Till skillnad från Stack/WrapPanel så anpassar sig inte Canvas efter vad som ligger i den och objekt som ligger i måste positioneras, vilket görs oftast med `Canvas.Left` och `Canvas.Top`. P.g.a. detta är Canvas endast användbart i undantagsfall när det gäller blanketter med flytande uppbyggnad, utan används främst i blanketter med fasta sidor (FixedPage) som det normala sättet att positionera objekt eller bara för att kodmässigt gruppera objekt i denna typ av blankett.

4.24.5 Ifyllnadsfält

4.24.5.1 *Generella XAML attribut*

Attribut utöver de nedan används normalt inte och behöver inte nödvändigtvis ha någon funktion i Forms, även om vissa säkert skulle fungera. De flesta attribut går också att använda på de flesta objekt. Vissa attribut som t.ex. FixedPage.Top/Left bör ej heller användas (detta kommer bara fungera om objektet ligger direkt under FixedPage). Andra som t.ex. OriginX/OriginY fungerar bara på vissa typer av objekt.

`x:Name` = Index för fältet. Får ej innehålla å/ä/ö (kan orsaka problem mot andra funktioner som använder sig av Name) eller många andra specialtecken. Understreck kan användas istället för mellanslag. Måste vara unikt i blanketten och används bl.a. av Träd, Math och Signeringsfunktionerna.
`Width` = Anger fältets bredd. Används Width så ska inte MinWidth eller MaxWidth användas då Width kommer att slå ut dessa två och fältet kommer att bete sig felaktigt i Forms

förutsatt att inte alla tre har exakt samma värde, vilket är lätt att missa. Width ska förutom i undantagsfall inte användas på CheckBoxar.
`MinWidth` = Anger fältets minsta bredd. Ska endast användas om fältet ligger i en WrapPanel. Tillåter fältet att variera sin storlek ner till MinWidth:s värde i en WrapPanel.
`MaxWidth` = Anger fältets största bredd. Ska endast användas om fältet ligger i en WrapPanel. Tillåter fältet att variera sin storlek upp till MaxWidth:s värde i en WrapPanel.
`Height` = Anger ett objekts höjd. Ska förutom i undantagsfall inte anges på några ifyllnasfält då dessa ska få sina höjder från textstorlekar och antal rader m.m.
`Canvas.Top` = Anger positionen i y-led utgående från Canvasens övre vänstra hörn. Bör inte användas i blanketter utan FixedPage eller på element som ligger i en Stack- eller WrapPanel då positionen är absolut. Objektet måste dock alltid ligga i en Canvas för att attributet ska gå att använd.
`Canvas.Left` = Anger positionen i x-led utgående från Canvasens övre vänstra hörn. Bör inte användas i blanketter utan FixedPage eller på element som ligger i en Stack- eller WrapPanel då positionen är absolut. Objektet måste dock alltid ligga i en Canvas för att attributet ska gå att använd.
`Style` = Vilken stil som ska användas för fältet och standardstilarna för TextBox. Stilar definieras annars i StackPanel.Resources eller FixedPage.Resources. En ”bugg” i Forms gör dock att textboxars och checkboxars Style attribut återställs när fält av- och på- markeras samt när blanketten skrivs ut i Forms.
`Grid.Column` = Styr i vilken kolumn i en Grid som objektet placeras i. Värdet 0 är kolumn ett. Anges inget placeras det i kolumn ett.
`Grid.Row` = Styr i vilken rad i en Grid som objektet placeras i. Värdet 0 är rad ett. Anges inget placeras det i rad ett.
`Tag` = Bör skrivas som ett separat underelement till dess fält för att göra den mer synlig och underlätta justering. Tag attributet innehåller gentemot Forms fältets xPath, alltså länken till XML filen som styr var fältets data ska spara i databasen. Här placeras också datatypkonverterare beroende på vilken datatyp som står angiven i XSD och ev. tillhörande konverteringsregel samt övriga valideringsfunktioner. `Tag` skrivs i formatet `|xPath|Converter|Parameter|Rule|Valideringsregler`, mer om detta går att läsa i **Beskrivning av funktioner i Forms**.
`Margin` = Används för att skapa mellanrum mellan objekt som ligger Stack-/WrapPanel och anges i formatet: ”left, top, right, bottom”.
`HorizontalAlignment` = Styr var i horisontalled som underelement bl.a. i en StackPanel eller Grid placeras. Fungerar ej på element i en WrapPanel. `FlowDirection` kan också användas men alla objekt kommer då också i omvänd ordning.
`VerticalAlignment` = Styr var i vertikalled som underelement bl.a. i en WrapPanel eller Grid placeras. För att detta ska fungera måste det finnas ett element i WrapPanel:n som fyller ut hela dess höjd. Man måste ibland använda en kombination av StackPanel och WrapPanel som underelement till varandra med respektive alignment och ordning för att få objekt att hamna exakt som man vill.
`ToolTip` = ToolTip på ett objekt. Radbryt i text kan t.ex. göras genom något liknande
`<TextBox.ToolTip><TextBlock>Line1<LineBreak/>Line2</TextBlock></TextBox.ToolTip>`

`FontFamily` = Vilket typsnitt som ska användas. Antingen kan systemtypsnitt användas som `Segoe UI`, `Times New Roman`, `Arial`, `Arial Bold`, `Broadway` osv. Eller så kan man använda de fonter som följer med i Forms i resources mappen genom att skriva t.ex. `/resources/Fonts/#Times New Roman Bold` (för filen timesbd.ttf) eller `/resources/Fonts/#Code 128` (för filen code128.ttf). Man kan här också referera till filnamnet istället för typsnittsnamnet genom t.ex. `/resources/Fonts/timesbd.ttf`.
`FontSize` = Textstorlek i pixlar (px). I jämförelse med storlek i punkter (pt) som används i t.ex. Microsoft Word så skiljer det i skalan 1:1,3333. Detta betyder att 12pt = 16px, 10 pt = 13,3333 px, 8pt = 10,3333 px osv.
`Text` = Kan endast användas på fält som inte har någon xPath då dessa alltid skrivs över när en blankett öppnas i Forms. Genom att använda TextBlock.Text kan man lägga in fast text i blanketten istället för att ha denna i Glyphs. Att i Visual Studio använda Text på ett TextBlock med xPath kan man visa positionen på denna, eller förhandsvisa hur texten kommer att bli då Forms ändå alltid skriver över denna när blanketten öppnas. Vill man ha förifyllda värden i ett fält med xPath finns det 3 sätt att göra detta: i Connectorn, i EBIM.xml (RoadMap) och i XML-filen.

Ytterligare information om attribut finns nedan för specifika fälttyper, dessutom finns utökade regelverk, hur man får/inte får använda attribut m.m. i **03b - Regelverk för Forms.docx**.

4.24.5.2 *Textboxar*

```
<TextBox x:Name="TextBoxName" Style="{DynamicResource tbStyle}">
    <TextBox.Tag>|xPath|stringEncodedConverter|</TextBox.Tag>
</TextBox>
```

4.24.5.2.1 Attribut

`TextWrapping` = Styr om text får fortsätta på kommande rader eller inte.
`AcceptsReturn` = Styr om användaren får göra enterslag.
`MinLines` = Minsta antal rader. Får aldrig vara större än MaxLines. Bör alltid vara lika med MaxLines om textboxen inte ligger direkt i en StackPanel eftersom det senare krävs för att den ska kunna ändra antalet rader som visas i Forms.
`MaxLines` = Största antal rader. TextWrapping = Wrap och Acceptsreturn = True krävs för att ha ett värde större än 1, annars kommer den att ta upp fler rader utan att de går att använda.

4.24.5.3 *Textblock*

```
<TextBlock x:Name="TextBlockName" Style="{DynamicResource tblStyle}">
    <TextBlock.Tag>|xPath|stringEncodedConverter|</TextBlock.Tag>
</TextBlock>
```

TextBlock är den enda typen av ifyllnadsfält som får användas utan någon Tag och xPath, eftersom det är ett fält där användaren inte kan påverka innehållet och därför inte kan skriva information som försvinner. TextBlock utan Tag attribut kan därför användas för att visa fast text på samma sätt som en Glyphs. Skulle man ha någon annan typ av infyllnadsfält utan xPath så ha den ingen stans att spara sin information och skulle användaren då skriva något eller göra ett val så kommer detta inte finnas kvar när man stänger blanketten.

4.24.5.3.1 Attribut

`TextWrapping` = Styr om text får fortsätta på kommande rader eller inte. Detta kräver att en fast bredd sätts på textblocket annars kommer det fortsätta väga åt vänster ut utanför blankettens kant.
`Tabindex` = Får ej användas på TextBlock.
`MinLines` = Minsta antal
`Margin` = För att få texten i ett TextBlock i nivå med det i en TextBox ska margin vara `"3,1,0,4"` istället för dess standardvärde på `"4,0,0,4"`. Det ska dock noteras att om flera TextBlock ska komma efter varandra i en WrapPanel så är ett standardmellanslag 4 och då ska `"3,1,0,4"` användas istället.

4.24.5.4 *Checkboxar*

```
<CheckBox x:Name="CheckBoxName" Style="{DynamicResource cbStyle}">
    <CheckBox.Tag>|xPath|Converter|ConverterParameter</CheckBox.Tag>
</CheckBox>
```

Det värde som sparas ner i databasen är det värde som står i CheckBoxens ConverterParameter när just den är vald.

4.24.5.4.1 Attribut

`Width` = Styr om text får fortsätta på kommande rader eller inte.
`AcceptsReturn` = Styr om användaren får göra enterslag.
`MinLines` = Minsta antal

4.24.5.4.2 Enval

För att CheckBoxar ska fungera som enval ska de alla ha samma xPath, men sedan olika parametrar.
Ska max två alternativ finnas att välja på, t.ex. Ja/Nej, så kan dess xPath vara kopplad mot ett Boolean värde. Convertern måste vara stringEnumBooleanConverter och de måste ha olika värden som parametrar, är de kopplade till ett Boolean värde kan bara 1 och 0 användas som parametrar.

```
<CheckBox x:Name="Ja" Style="{DynamicResource cbStyle}">
    <CheckBox.Tag>|xPath|stringEnumBooleanConverter|1</CheckBox.Tag>
</CheckBox>
<CheckBox x:Name="Nej" Style="{DynamicResource cbStyle}">
    <CheckBox.Tag>|xPath|stringEnumBooleanConverter|0</CheckBox.Tag>
</CheckBox>
```

För att ha fler val än två så måste de vara kopplade mot ett String värde (normalt String50). Använder man ett String värde kan man även ha text som parametrar (även å/ä/ö och mellanslag).

```
<CheckBox x:Name="Ja" Style="{DynamicResource cbStyle}">
    <CheckBox.Tag>|xPath|stringEnumBooleanConverter|Ja</CheckBox.Tag>
</CheckBox>
<CheckBox x:Name="Nej" Style="{DynamicResource cbStyle}">
    <CheckBox.Tag>|xPath|stringEnumBooleanConverter|Nej</CheckBox.Tag>
```

```
</CheckBox>
<CheckBox x:Name="Kanske" Style="{DynamicResource cbStyle}">
    <CheckBox.Tag>|xPath|stringEnumBooleanConverter|Kanske</CheckBox.Tag>
</CheckBox>
<CheckBox x:Name="Kanske inte då" Style="{DynamicResource cbStyle}">
    <CheckBox.Tag>|xPath|stringEnumBooleanConverter|Kanske inte
då</CheckBox.Tag>
</CheckBox>
```

4.24.5.4.3 Flerval

Flerval är helt enkelt CheckBoxar som är enskilda (de måste ha unika xPath).
Man kan om man använder ett Boolean värde som xPath använda BooleanConverter och behöver då inte ha någon parameter eftersom BooleanConvertern sköter det själv.

```
<CheckBox x:Name="Ja" Style="{DynamicResource cbStyle}">
    <CheckBox.Tag>|xPath|BooleanConverter|</CheckBox.Tag>
</CheckBox>
```

4.24.5.5 *ComboBox*

Med en ComboBox kan man få dropdownlist-funktionalitet i en blankett. Genom att placera en text som ett ComboBoxItem så kommer det värdet att sparas ner till ComboBoxens xPath. Detta går antinget att göra med ett TextBlock eller bara en Text. Med hjälp av `TextSearch.Text` så går det att sätta ett annat ”value” på Item än dess content för att sparas till XML:et.
ToDo: Testa om det går att använda Tag på textboxarna.

```
<ComboBox x:Name="ComboBox" MinWidth="80">
  <ComboBox.Tag>|xPath|stringEncodedConverter|</ComboBox.Tag>
  <ComboBoxItem TextSearch.Text="A">
    <TextBlock Text="Val1"></TextBlock>
  </ComboBoxItem>
  <ComboBoxItem TextSearch.Text="B">
    <TextBlock Text="Val2"></TextBlock>
  </ComboBoxItem>
  <ComboBoxItem>
    Val3
  </ComboBoxItem>
  <ComboBoxItem Content="Val4"/>
</ComboBox>
```

4.24.5.6 *Övriga fälttyper*

Förutom hittills nämnda layout objekt och ifyllnadsfält finns även ytterligare objekt i XAML som kan användas. Detta är en beskrivning av några av dem.

4.24.5.6.1 Hyperlink

Kan användas för att skapa länkar och placeras alltid i ett textblock. Det går sedan att länka till en URI, t.ex. en sida eller fil på internet eller på ett intranät. Länken kommer att öppnas i ett nytt fönster när man klickar på den i Forms.

```
<Hyperlink TargetName="_blank"
NavigateUri="http://utv.gainit.se/DDM_test/EyeDocWsA/EyeDoc_NET/ED_Download
```

```
.aspx?BlankettId=3820&amp;FilerId=10737">patientinformationsblad</Hyperlink
>
```

4.24.6 Styles

(EB, 2014-08-07, Forms v1.1.0.330)

4.24.6.1 *Formsrecourcedictionary*

Filen `formsresourcedictionary.xaml` innehåller några av de grundstilar som används som standard i Forms. Följande kod kan placeras i rotelementets `<StackPanel.Resources><ResourceDictionary>` för att ge ifyllnadsfälten samma utseende i Visual Studio som de kommer ha i Forms och därför göra det lättare att layouta blanketten:

```
<ResourceDictionary.MergedDictionaries>
  <ResourceDictionary
Source="/Resources/Dictionary/formsresourcedictionary.xaml"/>
</ResourceDictionary.MergedDictionaries>
```

Denna kod behövs inte för Forms utan kan plockas bort när blanketten är klar, till skillnad från tidigare skapar det dock inte längre några kända problem att ha det kvar vid publicering.

4.24.6.2 *Egendefinierade stilar*

För att ändra utseende på text och andra objekt och man vill kunna återanvända dessa kan man lägga in dem som definierade stilar i blanketten. Dessa stilar placeras i en `<ResourceDictionary>` till ett element och kan sedan användas av alla underliggande objekt av den specificerade typen, normalt placeras de i den yttre StackPanel:n för att sedan kunna användas i hela blanketten.
I XAML finns ett stort antal stilegenskaper som kan sättas till en stil, dessa kommer från System.Windows.Style (http://msdn.microsoft.com/en-us/library/system.windows.style.aspx), ett par enklare exempel visas nedan.

```
<Style x:Key="pageHeader" TargetType="TextBlock">
  <Setter Property="FontSize" Value="20"/>
  <Setter Property="FontFamily" Value="Segoe UI"/>
  <Setter Property="FontWeight" Value="Bold"/>
</Style>

<Style x:Key="buttonPMDoc" TargetType="Button">
  <Setter Property="Opacity" Value="0.7"/>
  <Setter Property="Background" Value="Transparent"/>
  <Setter Property="VerticalAlignment" Value="Center"/>
  <Setter Property="Height" Value="20"/>
  <Setter Property="BorderThickness" Value="0"/>
  <Setter Property="Cursor" Value="Hand"/>
</Style>
```

För att koppla stilen till ett objekt används sedan dess Style-attribut enligt:

```
<TextBlock Style="{DynamicResource pageHeader}"/>
```

```
<Button Style="{DynamicResource buttonPMDoc}"/>
```

4.24.6.3 *Validation.ErrorTemplate*

I äldre blanketter finns ibland Validation.ErrorTemplate på fält som har Rule i sin Tag för att kunna markera fält som har valideringsfel. Detta är dock förlegat då det idag hanteras av Forms direkt och kan plockas bort från blankettens fält. Det skapar inga problem att ha det men är helt enkelt onödigt.

Attributet på elementet ser ut enligt:

```
Validation.ErrorTemplate="{DynamicResource validationTemplate}"
```

"Stilen" såg sedan ut enligt nedan:

```
<ControlTemplate x:Key="validationTemplate">
  <Border BorderThickness="1" BorderBrush="Red" CornerRadius="2">
    <AdornedElementPlaceholder />
  </Border>
</ControlTemplate>
```

4.25 Kopplingar till externa system

Messenger, EID, Kuvert, ICD10.

5 Databasbeskrivning

Forms är enbart en gränssnittsmodul och har ingen egen kontakt med databas. All databashantering sköts av respektive modul som Forms har kontakt med vissa av dessa direkt eller indirekt via Kuvertet.

6 Teknisk plattform

6.1 System och plattformskrav

På alla klienter som ska köra applikationen:

- .NET Framework v2.0, v3.5 samt v4.0.
- Webbläsare med stöd för XBAP och internetinställningar som tillåter XBAP att laddas ner, installeras och köras.
- Serverns URL måste ligga som trusted site i internetinställningar eller ha en säkerhetsnivå av högst medel eller lägre.

7 Definitioner och förkortningar

Ord	Beskrivning
SVP	Standardvårdplan