|  |  |
| --- | --- |
|  | **2015** |
|  | CARMAK AB  Lars-Göran Andersson |

|  |
| --- |
| **[ FastReport implementation i befintligt system VIS/VIDA]** |
| Specifikation och design av implementering av rapportsystemet FastReport |

Innehåll

[Förutsättningar 2](#_Toc413152689)

[Tabeller 3](#_Toc413152690)

[GIT 3](#_Toc413152691)

[Kravställning 4](#_Toc413152692)

[Krav från Applikation 4](#_Toc413152693)

[Krav från underhållsapplikation 4](#_Toc413152694)

[Design 5](#_Toc413152695)

[Förutsättningar 5](#_Toc413152696)

[Användarinterface (GUI) 5](#_Toc413152697)

[Allmänt 5](#_Toc413152698)

[Rapportträd 5](#_Toc413152699)

[Kommandofönster 5](#_Toc413152700)

[Data, Datastrukturer 6](#_Toc413152701)

[Allmänt 6](#_Toc413152702)

[Rapportkomponenter 6](#_Toc413152703)

[Databas 6](#_Toc413152704)

[Parametrar 7](#_Toc413152705)

[Diagram 9](#_Toc413152706)

[Installation 20](#_Toc413152707)

[Allmänt 20](#_Toc413152708)

[Bibliotek 20](#_Toc413152709)

[Tabeller 20](#_Toc413152710)

[Lagrade Procedurer 20](#_Toc413152711)

FastReport implementation i befintligt system VIS/VIDA

# Förutsättningar

Befintlig databas: MSSQL

Databasen innehåller Tabeller, vyer och lagrade procedurer som förutom Applikationsdata även hyser tabeller för konfiguration av rapportsystemet.

Rapportsystemet konfigureras för:

* Identifiering av rapport (Rapportnummer vs Rapport-template .fr3-fil)
* Databaskopplingar per rapport (Endast lagrade procedurer används för dataåtkomst)
* Kriterier för val av rapport (Klient, Roll och Dokumenttyp)

För att klara en mjuk övergång från CrystalReport till FastReport skapas nya tabeller vid behov så att systemen kan samexistera utan problem. Subrapporter måste få egna lagrade procedurer för just FastReport då parametrar måste anges separat till de lagrade procedurerna vilket i CrystalReport sker internt.  
Då huvudrapporter använder Tabeller istf. Lagrade procedurer måste nya lagrade procedurer skapas som ersättning.

Applikationerna utrustas med valmöjlighet för växling mellan de olika rapportsystemen ONLINE.

Ett grafiskt underhållsprogram skapas för att underhålla konfigurationen av FastReport.  
Programmet ges namnet FastReport.

Applikationsinterfacet mot FastReport skall även användas av underhållsprogrammet.

Underhållsprogrammet skall även kunna användas för aktivering av FastReports designeditor för att skapa och ändra rapporter. Dessutom skall det vara möjligt att testa rapporterna för förhandsgranskning, utskrift och PDF-generering.

## Tabeller

Ursprungliga:

* Reportnames
* ClientPrefDoc
* DBProps

Nya

* FastReportNames - Nya datafält behövs för lagring av bl.a. Lagrade Procedurer
* FastSubReportNames Lagring av Subrapporter med Lagrade procedurer etc.
* DocType Koppling mellan nr och namn

Inga förändringar i ClientPrefDoc, kopplingen till rapportnr ges via nycklarna: ClientNo, RoleType och DocType.

DBProps: Nytt fält för lagring av sökväg till rapport-templates. Kan göras utan påverkan på befintligt system.

ReportNames: Här sker så mycket med data att det är säkrast att skapa en separat tabell för FastReport.

FastSubReports: Finns ej sedan tidigare. Behövs för att knyta underrapporter till Huvudrapporten och för att speca lagrad procedur till underrapporten. Vi lagrar även namnet på dataset som FastReport utnyttjar för DB-access. Detta visas i fönstret för rapport design.

Vid testkörning av rapporter bör också tilldelning av parametervärden vara möjlig. Det behövs således en funktion som kanreturnera en lista på parametrar som kan tilldelas värden för aktuell rapport.

## GIT

Versionshantering och backup av källkod hanteras med hjälp av GIT

Repositoryt för backup ligger på <https://github.com/CarmakAB/>

Gemensamma filer för applikationer finns i mappen ”FastReport” under projektet CommonSources.

Underhållsapplikationen finns i projektet FRMaintainer

# Kravställning

### Krav från Applikation

Systemet skall kunna generera rapporter med möjlighet till:

* Förhandsgranskning
* Utskrift
* Skapande av PDF-filer
* Hantera text-data i RichText-format.

Ett enkelt anrop skall räcka för att starta rapporten.

Anropet sker med parametrar för:

* Antal kopior (0 – anger att antalet kopior tas från filen ClientPrefDoc)
* ClientNo – nyckel till rapportval
* RoleType – nyckel till rapportval
* DocType - nyckel till rapportval
* Parameterlista med parametervärden
* Typ av output (Förhandsgranska, Utskrift eller skapande av PDF-fil)

### Krav från underhållsapplikation

Rapporten skall kunna aktiveras direkt genom att välja önskad rapport ur en lista på samtliga rapporter.

För vald rapport skall aktuella parametrar listas för värde sättning.

Rapport skall kunna:

* skrivas ut
* förhandsgranskas
* lagras som pdf-fil

Nya rapporter skall kunna läggas upp (Konfigurering och Design)

Rapporter skall kunna omkonfigureras och re-designas

Rapporter skall kunnas tas bort( Vi tar bara bort konfigureringen – templaten blir kvar)

# Design

## Förutsättningar

Utvecklingsmiljö: DELPHI XE6

Rapportprogramvara: FASTREPORT 4

## Användarinterface (GUI)

### Allmänt

Användarinterfacet skall ge användaren en överskådlig bild av samtliga rapporter som ingår i systemet.  
Användaren skall genom denna bild kunna välja önskad rapport för redigering, och test av de olika rapporteringssätt som systemet tillhandahåller, t.ex. Förhandsgranskning, utskrift och rapportering på PDF-fil etc.

### Rapportträd

Rapporterna visas upp i en trädstruktur där rapporterna grupperas efter dokumenttyp (DocType).

Rapportens ikon visar om konfigureringen saknar data eller är felaktig.

Högerklick på en rapport visar en popup-meny med olika kommandon

Vid förändringar i rapporter uppdateras rapportträdet genom att tabellen ”FastReportNames” läses in och bygger upp rapportträdet på nytt.

### Kommandofönster

Aktiviteter som kan utföras på markerad rapport visas upp i form av knappar, en knapp för varje aktivitet.

* Skapa ny rapport
* Redigera rapport (ej design)
* Designa rapport
* Förhandsgranska
* Skriv ut
* Spara rapport som PDF-fil
* Ta bort Rapport

## Data, Datastrukturer

### Allmänt

Nya typer som deklareras i systemet ges prefixet: ’TCM’ för att inte kollidera med andra typer som kan förekomma. ’CM’ står för: (C)ar(M)ak.

### Rapportkomponenter

Vi utgår från nödvändiga komponenter i FR4.biblioteket:

Endast fyra komponenter behövs för att uppfylla ställda krav:

1. frxReport
2. frxDBDataset
3. frxPDFExport
4. frxRichObject

**frxReport** är själva rapporthanteringen med utskrifter etc.

**frxRichObject** är en komponent som hjälper fxReport att hantera text/data som innehåller RichText-formaterad text.

**frxPDFExport** är en komponent som möjliggör att rapporter kan sparas som PDF-filer.

**frxDBDataset** kopplar datakällor till rapportkomponenten. Flera dataset kopplas automatiskt till rapporten allteftersom man skapar dem om de har skapts med samma formulär som ”owner”.  
OBS! Varje dataset måste ha ett unikt namn i gruppen. Detta fixas genom att låta namnet bildas av ’DS’+Underrapportens namn och för Huvudrapporten kalla den ’DsmainRep’.

### Databas

Två tabeller används för lagring av information för respektive rapport.

1. FastReportNames Huvudrapport
2. FastSubReportNames Underrapport

En rapport identifieras av ett Rapportnummer som även gäller för underrapporter.  
En underrapport har dessutom ett löpnummer som gör det möjligt att ha flera underrapporter under samma huvudrapport.

Både huvudrapporter och underrapporter använder sig av lagrade procedurer för databasåtkomst. Endast en lagrad procedur per rapport kan användas, dvs 1 för huvudrapport och 1 för varje ingående underrapport. Rapportkomponenten **frxReport** använder sig av komponenten **frxDBDataset** för att koppla sig till de lagrade procedurerna. En datasetkomponent för varje lagrad procedur.

När en rapport skall köras hämtas rapportnamnet från tabellen ”ClientPrefDocFR” som nycklas av fälten:

* ClientNo
* RoleType
* DocType

En procedur i datamodulen dmFR vid namn getClientDocPref returnerar förutom rappornamnet även saker som:

* Antal Kopior
* Begäran om printerdialog (PromptUser)
* Kollatering (Collate)

Vilka har betydelse endast om rapporten skall skrivas ut.

Tabellen DocType används endast av underhållsprogramet för att kunna gruppera rapporter av likartad typ.  
Detta har nu visat sig vara ett problem då vissa rapporter måste vara ensamma om en Doctype för att kunna identifieras unikt via clientPrefDoc-modellen. Detta bör lösas på något sätt framöver.  
  
Jag tycker att man lägger till Raportnummer som parameter till proceduren getClientPrefDoc och på så sätt kunna behålla DocType parameterns funktion som gruppsamlande parameter.

Vi behöver då också lägga till ett fält för rapportnummret i tabellen ClientPrefDoc. Detta innebär att vi bör välja tabellkopian ClientPrefDoc FR för att undvika problem med användningen av CrystalReport under övergångstiden.

Vid skapande av nya rapporter används den lagrade proceduren Vida\_GetMaxNo för att erhålla nya rapportnummer.

### Parametrar

Parametrar som rapporten behöver läses från systemtabellen – ”information\_schema.parameters”.

Med följande fråga:

select PARAMETER\_NAME, DATA\_TYPE from information\_schema.parameters where specific\_name=:SP\_NAM

Resultatet lagras I en datastruktur av typ TDictionary som ingår I DELPHI-Sviten. Den är generisk och deklareras här som: TCMParamsInfo: TDictionary<string,string> och enskilt element som TCMParamInfo: Tpair<string, string>. Den första strängen sk. ”KEY” och består av parameterns namn t.ex. ’@INVOICENO’. Den andra strängen identifierar parameterns typ t.ex. ’int’, ’varchar’ etc.

Parametrar med värde lagras i en enkel lista: TCMParams: Tlist<string,variant> som hanterar värden av olika typer.  
I applikationen använder man den på följande sätt:

Var Params: TCMParams;  
Begin  
 Params := TCMParams.create;  
 Params.Add(‘@INVOICENO’, 123456);  
 Params.Add(‘@OrderTextNo’, ‘F14/063456’);

Data från Rapport och subrapporter samlas i objekt som skapas från klasserna TCMMReportData och TCMSReportData för huvudrapport respektive subrapport.

TCMMReportData och TCMSReportData härstammar båda från klassen TCMReportData. som har information om Parametrar, lagrad procedur

### Diagram

#### Klasser i Rapportsystemet

Två databastabeller används för hela rapporten inkluderande underrapporterna.

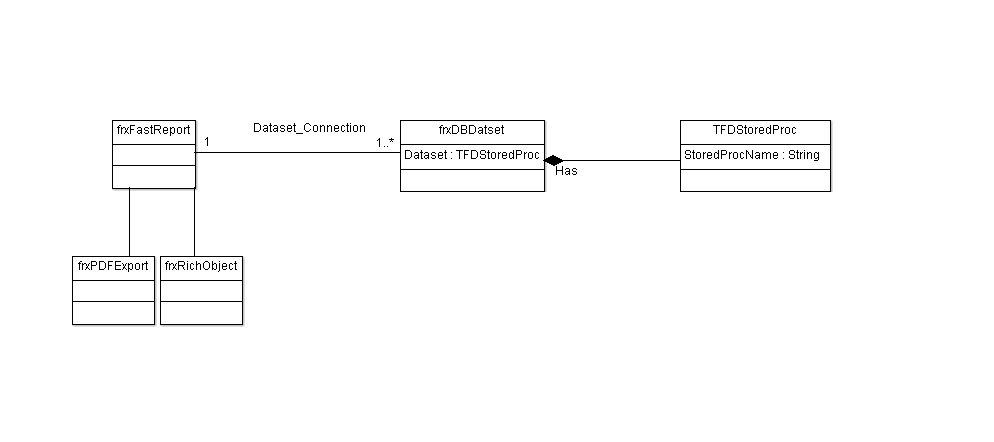
1. FastReportNames
2. FastSubreportNames

Gemensamt för tabellerna är att de håller namn på: lagrade procedurer och namn på Dataset som används av Rapport och underrapporter. Det gäller även plats för en kort beskrivning av rapport och underrapporter.

FastReportNames är till för huvudrapporten och har namnet på rapportfilen(Template/ReportName) som används. Namnet är fullständigt med extension(.fr3) men sökvägen ligger i en annan tabell(dbprops) gemensam för alla rapporter.  
FastSubReport har ingen rapportfil men refereras vid sitt namn i huvudrapporten, den har dock sin egen lagrade procedur och dataset.  
ReportNo eller rapportnummret knyter samman underrapporter och huvudrapport.

Klasserna TCMMReportData och TCMSReportData används endast som lagring av information om rapporter och tillhörande underrapporter.  
Vi har förutom Konstruktor, i klassen TCMMReportData också Destruktor som avallokerar alla data tillhörande Huvudrapporten och underrapporterna.  
TCMMReportData har också funktionen getAllParameters som returnerar en lista på samtliga parametrar en rapport behöver. Bra vid testkörning.

#### Rapportkomponenter



frxReport är den centrala rapportkomponenten och den har åtminstonde en komponent, frxDBDataset som kopplar sig mot databasen via ett dataset t.ex. en TFDStoredProcedure.

Det finns alltså en frxDBDataset-komponent för varje dataset som används. Man kan i princip ha hur många dataset som helst då komponenten frxReport hittar alla tillgängliga frxDBDataset komponenter.

I Detta system har vi valt att ha endast en **S**tored **P**rocedure som dataset till huvudrapporten och en **SP** per underrapport.

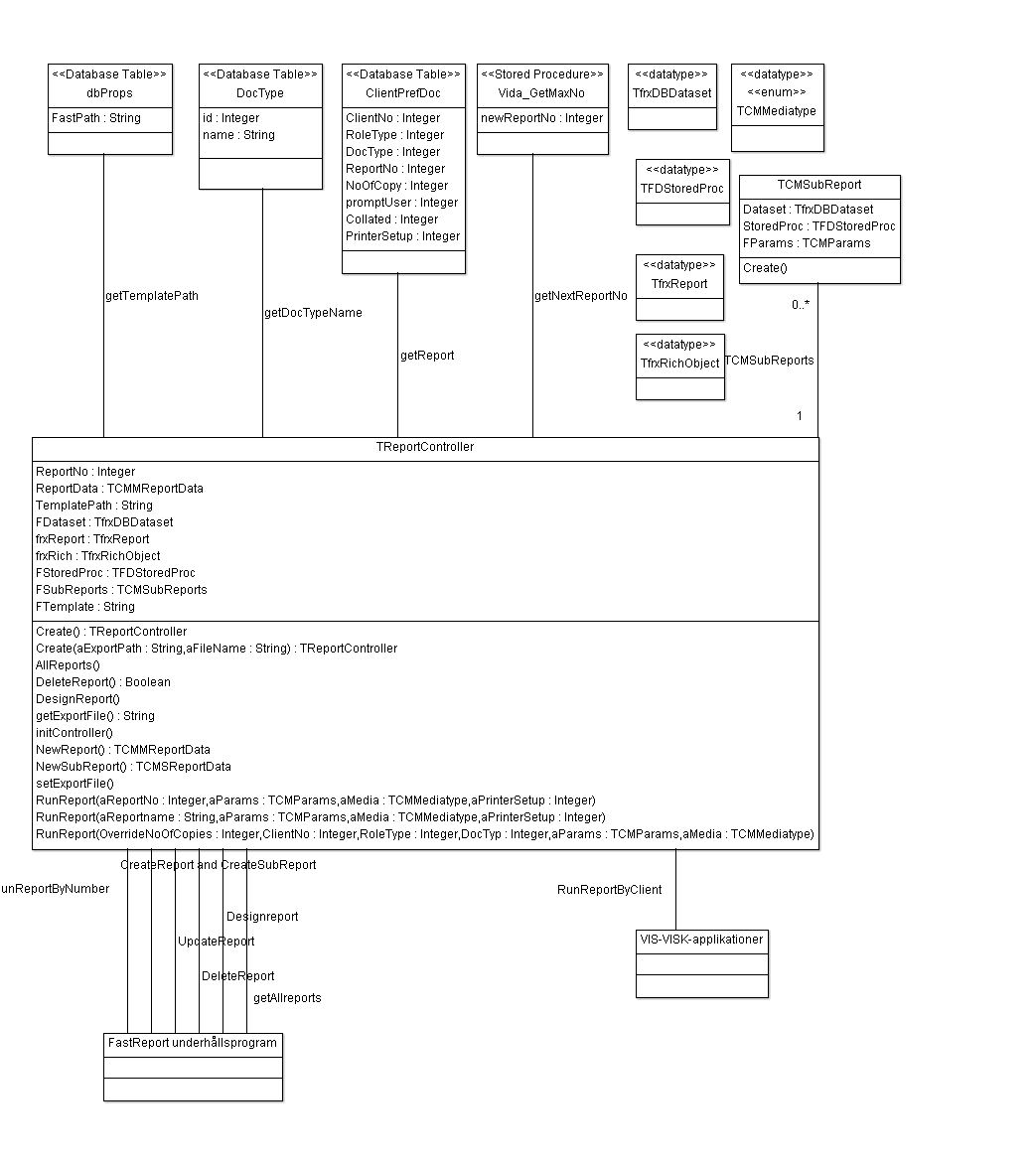
frxPDFExport-komponenten används för att generera en rapport på PDF-fil istf utskrift.

frxRichobject kopplas till fält som har sin text formatterad som RichText.

Alla dessa objekt skapas dynamiskt när en rapport skall skrivas ut.

Associationen ”Dataset\_Connection” sker automatiskt när ett nytt frxDBDataset-objekt skapas.

#### TCMReportController



Hjärtat i rapportsystemet där allt händer.

När en rapport skall skrivas ut, hämtas önskad rapport i form av rapportnummer från tabellen ”ClientPrefDoc”.

Nycklar för sökning är:

1. ClientNo
2. RoleType
3. DocType

Rapportnummer finns definierad som nyckel och skulle kunna ingå.  
Detta är intressant eftersom man helst skulle vilja undvika att använda Doctype som unik nyckel för specifika rapporter. Det spräcker iden med DocType men om man istället använde rapportnummret i sökningen så kunde rapporten behålla sin äkta DocType.

Via rapportnummret läses tabellerna ”FastReportNames” och ”FastSubReportNames” där namn på rapport-fil, lagrad procedur och datasetnamn för både huvudrapport och eventuella underrapporter.

Informationen lagras i klasserna TCMMReportData och TCMSReportData för huvudrapport respektive Underrapporter.

##### Några interna funktioner:

FetchReportData

Samlar in data för rapporten från databasen och bygger upp objekten ReportData och SubReportdata

PrepareReport

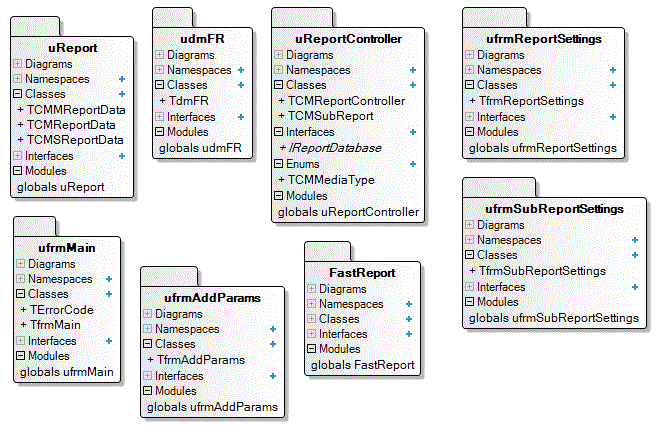
Skapar upp databasobjekten för Stored Procedures, frxDBDataset för huvud rapport och underrapporter. Detta sker i funktionerna :

* setupReport Anropar createDBComponents och   
   setupSubReports du underrapporter existerar.
* setupSubReports Anropar createDBComponents, skapar ett object av klassen TCMSubReportoch för varje existerande  
   underraport lägger upp en lista av dessa objekt som funktionen returnerar
* createDBComponents Skapar komponenter(objekt) för TFDStoredProcedure och frxDBDataset. SP:n prepareras och förses med parametrar och aktiveras. frxDBDataSet förses med ett unikt komponentnamn.

SetupfastReport

Skapar och initierar frxFastReport komponenten. Den skapar också frxRichObject komponenten.

#### Moduler



FastReport: Programmets ”FastReport” huvudmodul

ufrmMain: Huvudformuläret där all rapporthantering utgår ifrån. Dvs underhåll av rapportsystemet.  
Här finns trädet med alla rapporter, pop-up meny och knappar för skapande av nya rapporter, redigering, design etc.

ufrmReportSettings: Dialog där vald rapports egenskaper uppdateras/skapas.

ufrmSubReportSettings: Dialog där eventuell underrapports egenskaper uppdateras/skapas.

ufrmAddParams: Dialog där rapportens alla inparametrar presenteras för inmatning av data.  
Parametrarna samlas in genom att rapportens alla Lagrade procedurer undersöks med avsende på vilka parametrar de behöver. Underrapporternas lagrade procedurer ingår också. Inga dubletter sparas.  
Denna dialog används då rapporter skall testas för förhandsgranskning, utskrift eller skapande av PDF-fil. Även vid design används den då man kan förhandsgranska i designläget.

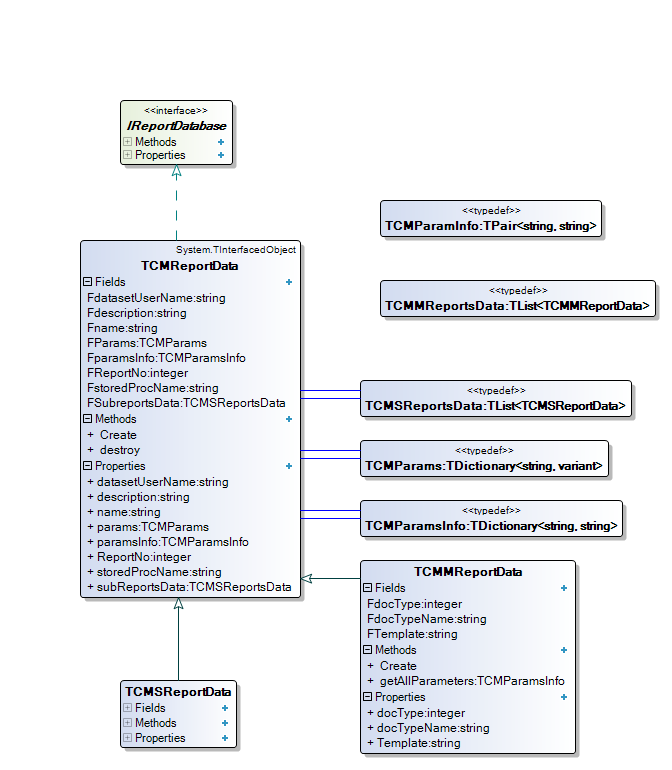
uReportController: Rapportsystemets hjärta. Här samlas data från databas och formulär, sammanställs i datastrukturer i form av klasser och listor av klasser. Modulen komunicerar med datamodulen udmFR som är interfacet mot databasen.  
Underhållsapplikationen skickar information om rapporter som här sammanställs och skickas vidare till datamodulen för uppdatering av databasen.  
Andra applikationer utnyttjar endast rapportgenereringen.  
Här har vi också en global variabel –useFR—som används i applikationerna för att kunna göra övergången från CrystalReports till FastReport så mjuk som möjligt. Den är bara ett sätt för applikationen att kunna kontrollera om FastReport skall användas eller inte och sätts normalt från programmens huvudmeny. Den har ingen funktion i FastReportsystemet.

uReport: Har klasser för Rapport och underrapport som samlar information om respektive rapport som behövs. Strukturer för listor av rapporter resp underrapporter ingår också. Ingen särskild bearbetning sker här.

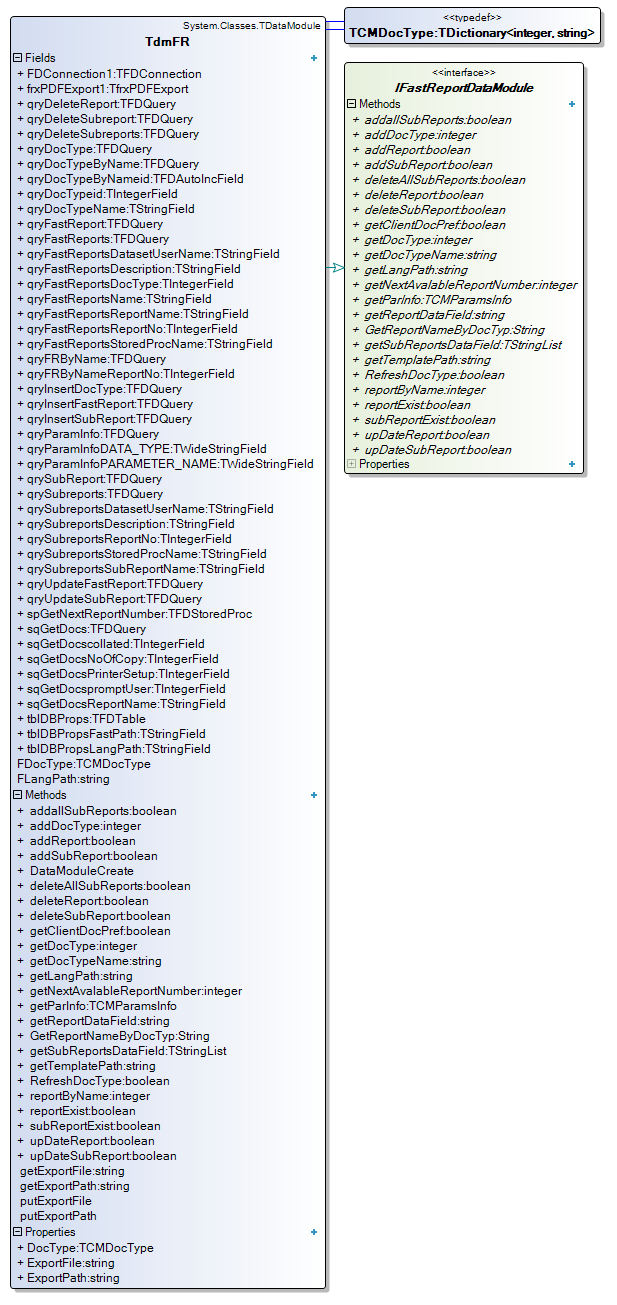
udmFR: Datamodulen där all kommunikation med databasen sker

##### Moduldiagram

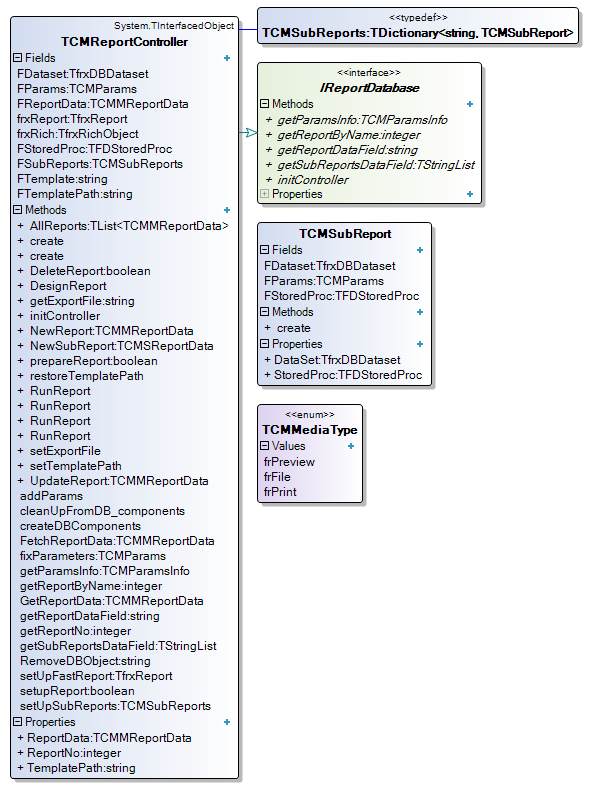
Modul: uReport



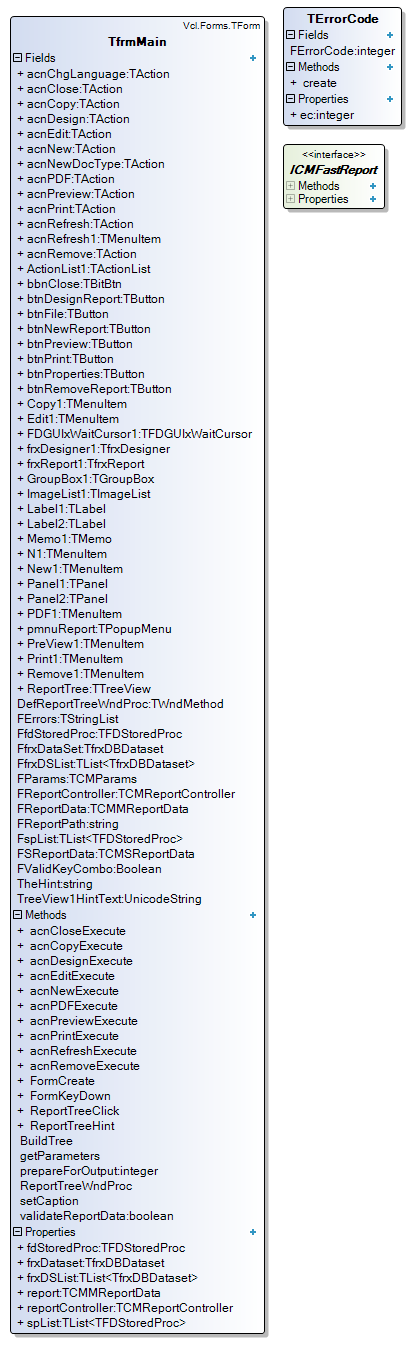
Datamodul: udmFR



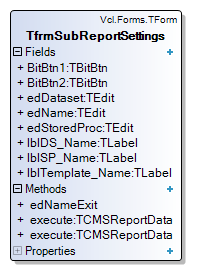
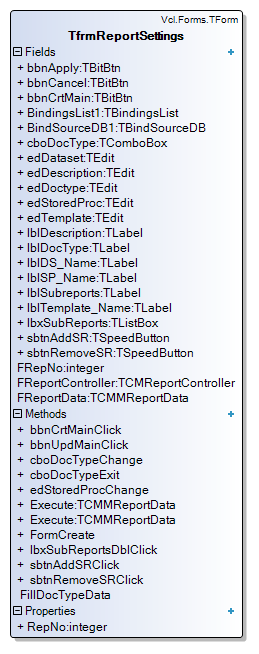
Modul: uReportController



Modul: ufrmMain



Moduler: ufrmReportSerttings och ufrmSubReportSettings



# Installation

## Allmänt

Systemet har testats på ALVESQL3 där också rapporter utvecklats.

Rapporterna har dock lagrats på ALVEFILE01 på samma plats som de gamla CrystalReports-rapporterna finns lagrade. De nya rapporterna har dock suffixet .fr3 istället för .RPT.

Allt detta innebär att här finns lagrade procedurer, nya tabeller och tabell innehåll redan klart och ska kopieras till driftsystemen för att undvika problem med olika namn och nummer.

När ”Testdatabasen” refereras, avses databasen ” vis\_vida” på ALVESQL03-servern.

## Bibliotek

Rapportfilerna läggs upp på ett gemensamt bibliotek, sökvägen till detta bibliotek läggs upp i tabellen **dbProps** i fältet ”FastPath”.

## Tabeller

FastReportNames och FastSubReportNames är nya tabeller

**DocType** är en ny tabell och innehållet kopieras från testdatabasen

Tabellen **dbProps** har fått ett nytt fält med namn ”FastPath” och är definierat som VARCHAR(255)

Tabellen **dbo.MaxKeyValue** skall uppdateras med en ny post för tabellen dbo.FastReportnames och initieras med ett maxvärde som högre än det högsta använda rapportnummret vid tillfället för installation.

## Lagrade Procedurer

Alla underrapporter har en egen lagrad procedur och deras namn har suffixet: \_FR

En del rapporter har vyer istället för lagrade procedurer och för dessa rapporter har nya lagrade procedurer skapats med suffixet: \_FR

###### Nya lagrade procedurer

|  |
| --- |
| Name |
| vida\_Adr\_FR |
| vida\_CMR\_Note\_FR |
| vida\_FS\_Note\_FR |
| vida\_FS\_Note\_INT\_FR |
| vida\_KONTRAKT\_Note\_FR |
| vida\_LO\_Note\_FR |
| vida\_LO\_Note\_INT\_FR |
| vida\_PkgSpec\_Note\_FR |
| vida\_TrpBrev\_Note\_FR |
| vida\_trporder\_note\_FR |
| Vis\_AOinput\_FR |
| Vis\_AOoutput\_FR |
| vis\_AVROP\_ALO\_Note\_FR |
| vis\_AVROP\_FS\_Note\_FR |
| vis\_AVROP\_LO\_Note\_FR |
| vis\_AVROP\_LO\_Note\_INT\_FR |
| vis\_AVROP\_PkgSpec\_Note\_FR |
| vis\_AVROP\_trporder\_Note\_FR |
| vis\_contract\_CertText\_FR |
| vis\_FSC\_StatExtInvNos\_FR |
| vis\_invoice\_CertText\_FR |
| vis\_PkgCodeLayout\_FR |
| vis\_PkgCodeSearchName\_FR |
| vis\_RawRun\_FR |
| vis\_ShipdAvrop\_FR |
| vis\_shipto\_FR |
| vis\_VerkGradProduct\_FR |
| vis\_VerkGradTop10\_FR |
|  |

###### Lagrade procedurer med parametrar

|  |  |  |
| --- | --- | --- |
| Name | PARAMETER\_NAME | DATA\_TYPE |
| vida\_Adr\_FR | @ShippingPlanNo | int |
| vida\_CMR\_Note\_FR | @OrderNo | int |
| vida\_FS\_Note\_FR | @OrderNoText | varchar |
| vida\_FS\_Note\_INT\_FR | @LONo | int |
| vida\_KONTRAKT\_Note\_FR | @ORDERNUMBER | varchar |
| vida\_LO\_Note\_FR | @OrderNoText | varchar |
| vida\_LO\_Note\_INT\_FR | @ShippingPlanNo | int |
| vida\_PkgSpec\_Note\_FR | @OrderNoText | varchar |
| vida\_TrpBrev\_Note\_FR | @OrderNoText | varchar |
| vida\_trporder\_note\_FR | @OrderNoText | varchar |
| Vis\_AOinput\_FR | @SortingOrderNo | int |
| Vis\_AOoutput\_FR | @SortingOrderNo | int |
| vis\_AVROP\_ALO\_Note\_FR | @ShippingPlanNo | int |
| vis\_AVROP\_FS\_Note\_FR | @ShippingPlanNo | int |
| vis\_AVROP\_LO\_Note\_FR | @ShippingPlanNo | int |
| vis\_AVROP\_LO\_Note\_INT\_FR | @ShippingPlanNo | int |
| vis\_AVROP\_PkgSpec\_Note\_FR | @ShippingPlanNo | int |
| vis\_AVROP\_trporder\_Note\_FR | @SHIPPINGPLANNO | int |
| vis\_contract\_CertText\_FR | @ORDERNUMBER | varchar |
| vis\_FSC\_StatExtInvNos\_FR | @StartYear | int |
| vis\_FSC\_StatExtInvNos\_FR | @StartMonth | int |
| vis\_FSC\_StatExtInvNos\_FR | @EndYear | int |
| vis\_FSC\_StatExtInvNos\_FR | @EndMonth | int |
| vis\_invoice\_CertText\_FR | @INVOICENO | int |
| vis\_PkgCodeLayout\_FR | @VariantNr | int |
| vis\_PkgCodeSearchName\_FR | @VariantNr | int |
| vis\_RawRun\_FR | @RunNo | int |
| vis\_RawRun\_FR | @RegPointNo | int |
| vis\_RawRun\_FR | @SupplierNo | int |
| vis\_ShipdAvrop\_FR | @OrderNo | int |
| vis\_shipto\_FR | @INVOICENO | int |
| vis\_shipto\_FR | @Reference | char |
| vis\_VerkGradProduct\_FR | @UserID | int |
| vis\_VerkGradTop10\_FR | @UserID | int |