Automatisk Webbaseret Opdateringssystem

Bo S. Christensen
Jacob R. Bentzen

Kongens Lyngby 2006
IMM-B. Eng. – 2006 – 46

Technical University of Denmark
Informatics and Mathematical Modeling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

Summary

This thesis documents the devolopment of an Automatic Webbased Updating system, developed in colaboration with Webtop Technology A/S.
The project is created as a final exam project for the completion of a degree as certified engineer in information technology at the Technical University of Denmark.
The project is developed over a period of 10 weeks, from 15th November 2005 to the 6th February 2006, with Bent Frøhlke Nielsen from IMM as supervisor.

The updating system is capable of keeping installations of products from Webtop Technology up to date via the Internet. Thereby easing the burden on the support department and adressing security problems, errors and other inappropriate issues.
Furtermore these updates could be giving extra value for the customers, by giving more features to their solution.

Resume

Denne rapport dokumenterer udviklingen af et Automatisk Webbaseret Opdatingssystem (AWO) for virksomheden Webtop Technology A/S.
Projektet er udarbejdet som afsluttende eksamensprojekt for Diplom-IT ingeniøruddannelsen ved Danmark Tekniske Universitet (DTU), over en periode på 10 uger, fra 14. november 2005 til 6. februar 2006, med Bent Frøhlke Nielsen som vejleder fra Institut for Matematiske Modelering.

Opdateringssystemet er i stand til automatisk at holde installationer af Webtop Technologys produkter opdateret via Internettet og dermed lette arbejdsbyrden for supportafdelingen og imødegå sikkederhedsproblemer, fejl og andre uhensigsmæssigheder. Samtidigt er opdateringerne med til at give Webtops kunder mere værdi af deres intranet, ved at introducere nye features.

Forord

Under udviklingen er der blevet anvendt Unified Process (UP) som udviklingsmetode og det forventes af læseren har kendskab til UML-diagrammer og systemudvikling generelt.
Rapporten omfatter de fire faser: Analyse, Design, Implementering og Test der beskriver udviklingsprocessen af systemet.

Systemet er udviklet i Microsoft .NET 2.0 frameworket, med Microsoft Visual Studio 2005 som udviklingsmiljø og med Microsoft Visual SourceSafe som dokumenthåndterings- og versioneringssystem

Vedlagt denne rapport

Den fulde kildekode til AWO-systemet er vedhæftet denne rapport på en CD-ROM.

Tak Til

Vi vil gerne takke Webtop Technology A/S for deres hjælp med tilblivelsen af dette projekt og for at stille udstyr, software mm. til rådighed.
Desuden vil vi gerne takke Jacob Schultz og Michael S. Christensen for korrektur-læsning og kommentarer til rapporten.

Indhold

KAPITEL 1

1. Projektdefinition

1.1. Problemstilling

Webtop Technology udvikler, implementerer og supporterer standardproduktet WebtopONE Corporate Portal, som primært bruges indenfor områder som Intranet, CMS og virksomhedsportaler.
WebtopONE produktsuiten består af en række moduler og komponenter, der hver dækker sit funktionalitetsområde.
Kravene til en sådan løsning ændrer sig med tiden og der opdages løbende uhensigtsmæssigheder eller fejl. Derfor findes hvert produkt i en række versioner. Disse versioner er løbende blevet installeret hos Webtops kunder, hvilket betyder at der eksisterer en mængde forskellige versioner installeret hos de forskellige kunder.

I dag bliver opdateringer annonceret på en supportportal, hvorefter kunderne kan hente rettelserne, eller bede supportafdelingen om hjælp til at få installeret disse.
Denne metode gør det sandsynligt at kunderne har forældede eller fejlbehæftede versioner i længere perioder. Dette giver samtidig en øget arbejdsbyrde i supportafdelingen.

1.2. Problemformulering

Der skal designes og udvikles en webapplikation, der kan holde kundernes WebtopONE produkter opdaterede.

Løsningen skal bestå af en central server, placeret hos Webtop, der er den centrale Versions Server.
På denne server skal være en service, som kunderne skal kunne koble op mod for at foretage opdateringerne via software på deres server.
Derudover vil der være software der skal installeres på Webtops server, der gør det muligt at vedligeholde Versions Serveren.

Det skal være muligt at:

- Administrere versionerne på Versions Serveren via et administrations interface.
- Downloade og installere de nye versioner via kundens WebtopONE portal, eller via en automatisk service, fx. en Windows Service, der selv kan checke efter nye versioner med et foruddefineret interval.
- Håndtere opdateringer fra en vilkårlig version til den nyeste.
- Håndtere begrænsninger i form af licenser, specialløsninger o.l.

Følgende skal undersøges i foranalysen:

- Kan Webtops nuværende installationspakker styres via denne løsningsidé (evt. konverteres) eller skal der udvikles dedikerede installere til dette formål?
- Kan installationsprogrammerne gennemføres uden at det kræver indtastninger eller øvrige påvirkninger af en bruger (Silent Install). Således at de oplysninger der skal bruges er standardværdier eller selv indsamles fra maskinen af installationsprogrammet.?
- Hvordan er det muligt at overføre installationspakkerne fra Versions Serveren til Kunde Serveren?
- Hvilket sprog eller framework skal bruges til at udvikle løsningen?

1.3. Projektafgrænsning

Webtop ønsker kun, at der er understøttelse for deres nye version 5.0 af deres WebtopONE produktsuite, da denne vil adskille sig væsentligt fra de tidligere versioner, bl.a. fordi den er den første version udviklet i .NET frameworket.
Da version 5.0 af WebtopONE ikke er færdigudviklet før dette projekt, vil løsningen være en prototype, der er baseret på de forventninger der er til den nye version.

KAPITEL 2

2. Definitioner og aktører

I rapporten er der en række definitioner og udtryk der vil blive brugt for at beskrive systemet. Derudover er der forskellige aktører med specifikke roller.
Disse definitioner og aktører er beskrevet herunder.

Administrations Komponenten: Den komponent på Webtop Serveren, der gør det muligt for en Webtop Administrator at vedligeholde Versions Serveren.
Aktiv Version: Version der kan opgraders, enten halv- eller fuldautomatisk.
AWO, Systemet, Løsningen: Synonymer for det produkt denne rapport beskriver.
AWOConfigurator: Applikation der gør det muligt at ændre opsætningen af Opdaterings Servicen, så som Kundens firmanavn, tidspunkt for fuldautomatisk opdatering, samt hvilke Produkter det er muligt at opdatere.
BITS: Background Intelligent Transfer Service, indbygget Windows service, som benyttes til overførsel af installationspakkerne.
Deaktivet Version: Version der kræver manuel tilpasning og det derfor ikke er muligt at opgradere til med AWO-systemet.
Fuldautomatisk opdatering: Opdatering der foretages på et foruddefineret tidspunkt af Opdaterings Servicen og udføres uden menneskelig indblanding.
GUID: Globally Unique IDentifier, et unikt 128 bit tal der kan bruges til identifikation. De kan generes af styresystemet eller applikationer på flere måder, men er gerne en kombination af flere unikke oplysninger, heriblandt tidspunktet for oprettelsen.
Halvautomatisk opdatering: Opdatering der kræver at en person starter processen, via Opdaterings Komponenten, før den påbegyndes.
Håndinstallation: Opdatering af et produkt der foretages manuelt, uden om denne løsning. Dette kan være nødvendigt i de tilfælde hvor selve opgraderingen ikke kan automatiseres til en færdig installationspakke.
IIS: Internet Information Services, gør det bl.a. muligt at benytte en maskine som webserver.
Kategori: Produktkategori der beskriver en samling af Produkter i AWO-systemet.

Kunde: Webtop kunde der har WebtopONE installeret.
Kunde Administrator: En person hos Kunden, der vedligeholder installationerne fra Webtop.
Kunde Serveren: Den server Kunden har sin WebtopONE installeret på.
Opdaterings Komponenten: Webbaseret GUI, som gør det muligt for en Kunde Administrator at foretage en halvautomatisk installation. Med halvautomatisk installation menes, at han selv kan starte installationen, hvorefter disse vil blive udført automatisk.
Opdaterings Komponenten har yderligere den samme funktionalitet som AWO-Configuratoren.
Opdaterings Service: En service der, som en del af løsningen, kører på Kundens Serveren. Servicen er i stand til at udføre fuldautomatiske opdateringer.
Produkt: Ethvert Webtop produkt, der kan opdateres af Systemet.
Produktoplysninger: Fællesbetegnelse for alle oplysninger om Produkter. Dette inkluderer både Kategorier, Produkter og Versioner.
Version: En specifik version af et Produkt.
Versions Serveren: Den centrale server, der indeholder Versionerne og informationer om disse.
Webtop: Webtop Technology A/S, virksomheden denne løsning er udviklet i samarbejde med.
Webtop Administrator: En person hos Webtop der vedligeholder Versions Serveren.
WebtopONE: Portalløsning hvor WebtopONE Corporate Portal Server, er kerneproduktet.
Webtop Serveren: Den server Webtop har sin WebtopONE installation på.

KAPITEL 3

3. Kravspecifikation

3.1. Introduktion

I samarbejde med Webtop Technology A/S er der blevet udviklet en kravspecifikation, med udgangspunkt i IEEE-830 standarden [IEEE-830], dog har vi fjernet de punkter vi ikke mente var relevante for dette projekt.

Denne kravsspefikation beskriver den produktionsklare udgave af produktet og indeholder krav der ikke er implementeret i dette projekt. Disse krav er behandlet i kapitel 9, Udvidelser.

3.1.1. Formål

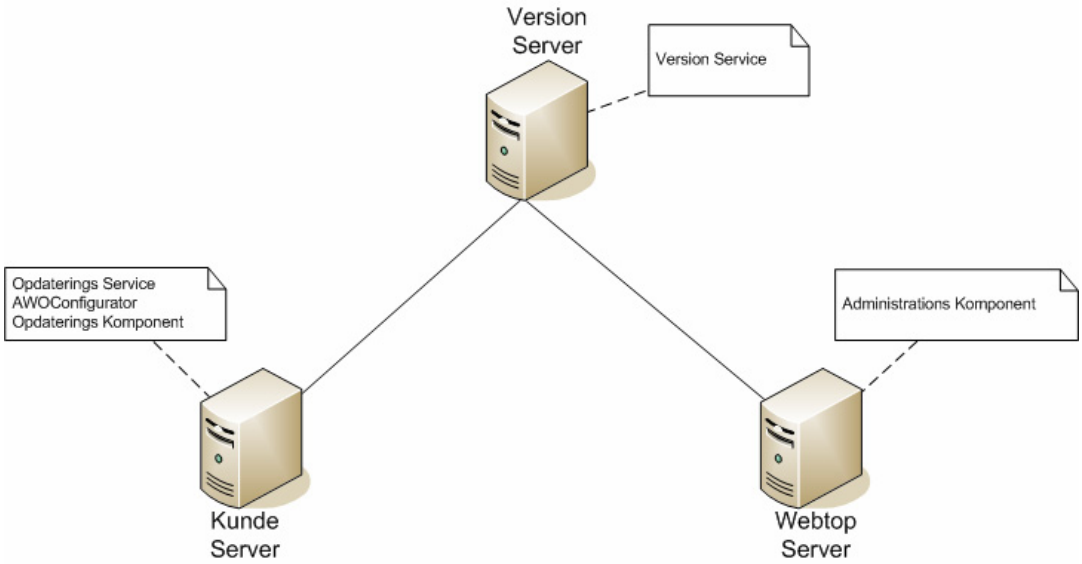
Denne specifikation beskriver kravene for et automatisk webbaseret opdateringssystem, udviklet specielt til WebtopONE produktsuiten.
Kravspecifikationen dokumenterer de krav der er aftalt i samarbejde med Webtop Technology A/S. Dokumentet danner også grundlag for designet af systemet.

3.1.2. Omfang

Systemet skal automatisk kunne opdatere Produkter fra WebtopONE produktsuiten hos Kunderne.

3.2. Overordnet beskrivelse af AWO

Webtop havde selv en ide til hvilke dele systemet skulle bestå af.
Tegningen herunder beskriver hvordan systemet fordeles på 3 servere, hvoraf Kunde Serveren og Webtop Serveren er eksisterende maskiner. Versions Serveren er en ny maskine, der skal administreres af Webtop.
Linierne giver udtryk for hvilke servere der skal kunne kommunikere med hinanden.



Figur 1: Delkomponenter og maskiner i systemet

3.2.1. Produktperspektiv

Systemet skal indgå som et uafhængigt add-on til WebtopONE produktsuiten. Opdaterings og Administrations komponenterne skal kunne indsættes på sammen måde, som andre WebtopONE komponenter på en portal.

3.2.2. Produkt funktioner

Systemet har 2 typer funktioner:

- Funktioner Kunden kan benytte:
- Funktioner Administratoren kan benytte.

Begge typer funktioner er beskrevet i sektion 3.3 af dette dokument.

3.2.3. Brugerkarakterstik

- **Kunde**: Webtop-kunde der har WebtopONE Corporate Portal Server, samt AWO installeret. De er allerede vant til at bruge WebtopONE produkterne. Da Opdaterings Komponenten har et interface der minder om de eksisterende WebtopONE komponenter. Ekstra oplæring i den nye komponent (Opdatering Komponenten), bør derfor være minimal.
- **Webtop Administrator**: Person der vedligeholder Versions Serveren, ansat af Webtop Technology. A/S. Administratoren er superbruger af WebtopONE produkterne. Det er acceptabelt, at der kræves oplæring for at bruge Administrations Komponenten.

3.2.4. Begrænsninger

Kunden kan ikke installere nye Produkter med denne løsning, da dette kræver køb af licenser o.l. Kunden har derimod mulighed for at installere opgraderinger via Systemet.
I nogle tilfælde kræver opgraderingen til en bestemt Version, at der foretages en række manuelle tilpasninger, der ikke kan inkorporeres i installationspakken.
Løsningen skal ikke kunne opgradere til en sådan version. Systemet skal dog kunne håndtere at sådanne versioner eksisterer, samt at disse ikke må kunne installeres automatisk.

3.2.5. Forudsætninger

- Kunden skal have WebtopONE Corporate Portal Server installeret for at kunne opgradere via Systemet.
- Webtop skal have WebtopONE Corporate Portal Server installeret for at kunne vedligeholde Versions Serveren..
- Det skal være muligt at benytte Windows Services med administrator rettigheder på Kunde Serveren.
- Der skal kunne skabes forbindelse mellem klienten og serveren. (Kunde Server til Versions Server, samt Webtop Server til Versions Server)
- Versionsoplysninger skal være tilgængelige hos Kunden.
- Kunden skal acceptere, at der sker kommunikation mellem deres intranet og den centrale Versions Server.

3.3. Specifikke krav

3.3.1. Opdaterings Servicen (Kunde Serveren)

Eksterne interfacekrav (Brugerinterfaces)
Da dette er en Windows Service, er der ikke noget brugerinterface.
Vigtig information om Servicen, så som exceptions og resultater for opdateringer, skal logges i Windows eventloggen.

Hardwareinterfaces
Opdaterings Komponenten skal kunne fungerer på hardware, der opfylder de minimumskrav der stilles til WebtopONE Corporate Portal server.
Disse krav er d.d.:

- CPU: Xeon TM - 1 processor på 2,0 GHz
- RAM: 1,0 GB
- Disk: 36 GB

Kommunikationsinterfaces
Opdaterings Servicen skal kunne kommunikerer med en WebService via HTTP-protokollen. Servicen skal yderligere kunne kommunikere med andre processer på samme server.

Funktionelle krav
Opdaterings Servicen skal kunne foretage fuldautomatiske opdateringer af Produkter, med foruddefineret faste intervaller.
Servicen skal gøre det muligt at foretage halvautomatiske opdateringer gennem Opdaterings Komponenten.

Performancekrav
Overførslen af opgraderingerne må ikke blokere for Kundens øvrige netværkstrafik.

Softwaresystemegenskaber
Den del af kommunikationen med Serveren, der indeholder fortrolige informationer, om Kunden skal krypteres.

Vedligeholdelse
Opdatering Servicen skal kunne opgradere sig selv, på samme måde som den kan med andre Produkter.

3.3.2. AWOConfigurator (Kunde Serveren)

Eksterne interfacekrav (Brugerinterfaces)
Brugerinterfacet skal så vidt muligt være intuitivt, men oplæring er dog acceptabelt i begrænset omfang.

Hardwareinterfaces
AWOConfiguratoren skal kunne fungere på hardware, der opfylder de minimums-krav der stilles til WebtopONE Corporate Portal server.
Disse krav er d.d.:

- CPU: Xeon TM - 1 processor på 2,0 GHz
- RAM: 1,0 GB
- Disk: 36 GB

Kommunikationsinterfaces
AWOConfiguratoren skal kunne kommunikerer med Opdaterings Servicen, der ligger på samme server.

Funktionelle krav
AWOConfiguratoren skal gøre det muligt, at foruddefinere, hvornår fuldautomatisk opdatering skal køres.
Det skal også være muligt, at fravælge opdatering af de enkelte Produkter. Dette kan f.eks. være, hvis en kunde har en specieludviklet udgave af et produkt, der derfor ikke skal opgraderes.
Det skal også være muligt at styre hvilke typer opdateringer der er tilladt.
Yderligere skal det være muligt for en Kunde Administrator, at ændre sine kundeoplysninger.

3.3.3. Opdatering Komponenten (Kunde Serveren)

Eksterne interfacekrav (Brugerinterfaces)
Opdatering Komponenten skal så vidt muligt have samme design som de øvrige WebtopONE komponenter.

Hardwareinterfaces
Opdatering Komponenten skal kunne fungere på hardware, der opfylder de minimumskrav der er stillet til WebtopONE Corporate Portal server.
Disse krav er d.d.:

- CPU: Xeon TM - 1 processor på 2,0 GHz
- RAM: 1,0 GB
- Disk: 36 GB

Kommunikationsinterfaces
Dataoverførsler skal kunne foregå via HTTP-protokollen.

Funktionelle krav
Opdatering Komponenten skal kunne præsentere Kunden for versionsoplysninger om de installerede WebtopONE produkter.
Opdatering Komponenten skal kunne opdatere et WebtopONE produkt til den nyeste Version, med begrænsningerne fra afsnit 3.2.4 taget i betragtning.

Performancekrav
Overførslen af opgraderingerne må ikke blokerer for Kundens øvrige netværks-trafik.
Kortvarige forstyrrelser og nedetid for portalen er acceptable under selve installationsprocessen, da dette normalt bør foregår når der er ingen, eller kun få brugerer på portalen.

Softwaresystemegenskaber
Den del af kommunikationen med Serveren der indeholder fortrolige informationer om Kunden, skal krypteres.

Vedligeholdelse
Opdatering Komponenten skal kunne opgradere sig selv, på samme måde som den kan med andre Produkter.

3.3.4. Versions Servicen (Versions Serveren)

Eksterne interfacekrav
Versions Servicen har i sig selv ikke noget GUI interface, men dens funktioner kan tilgås af andre applikationer

Kommunikations interfaces
Dataoverførsler skal kunne foregå via HTTP protokollen.

Funktionelle krav
Skal kunne servicere Opdatering Services og Administrations Komponenter med versionsoplysninger og installationspakker.
Versions Servicen skal kunne logge Kundernes installationer, samt hvilke opgraderinger der er påbegyndt og hvilke der er gennemført, for at lette fejlfinding o.l.

Softwaresystemegenskaber
Al kommunikationen der indeholder fortrolige oplysninger om Kunder skal krypteres.

Sikkerhed
Funktioner som kan hente oplysninger om kunder og ændre data på servicen, skal beskyttes, så disse kun kan bruges af Webtop administratorer.

Vedligeholdelse
Informationerne på Serveren bliver indsamlet via Opdatering Servicen og administreret via Administrations Komponenten.

3.3.5. Administrations Komponenten (Webtop Serveren)

Eksterne interfacekrav (Bruger interfaces)
Da denne komponent kun skal bruges af Webtop Administratorer, er det tilladt, at der er brug for oplæring før brug. Dog skal designet være så brugervenligt som muligt.

Kommunikationsinterfaces
Dataoverførsler skal kunne foregå via HTTP protokollen.

Funktionelle krav

- Administrations Komponenten skal gøre det muligt for en Administrator at tilføje, redigere og slette Versioner til Versions Serveren.
- Administrations Komponenten skal kunne præsentere oplysninger om de Versioner der er registreret på Versions Serveren.
- Administrations Komponenten skal kunne præsentere versions oplysninger om Kunderne for Administratoren.
- Administrations Komponenten skal gøre det muligt at se en Activity log, over påbegyndte og afsluttede opgraderinger.

Softwaresystemegenskaber

Den del af kommunikationen med Serveren, der indeholder fortrolige oplysninger om Kunden skal krypteres.

KAPITEL 4

4. Projektplanlægning

Vi har med stor succes tidligere anvendt Unified Process (UP), som udviklingsmetode for større IT-systemer og har derfor også brugt den i dette projekt.
Valget af UP er baseret på nedenstående betragtninger:

- Iterativ og inkrementiel udvikling, hvilket giver mulighed for at vurdere systemet i helhed og foretage eventuelle tilpasninger af designet, efter hver iteration.
- Bruger Objekt Orienteret Analyse og Design (OOA/D)
- UML som notationsform giver standardiserede figurer som er lette at læse
- Kontinuerlig test og revision.
- Nemt for Webtop og vejleder at følge udviklingen, da der er fungerede kode efter hver iteration.

Fra UP har vi valgt at bruge følgende discipliner og artefakter:

- Kravsstyring:
  - Projektbeskrivelse
  - Aktørliste og Use case diagrammer
- Analyse og Design
  - Klasse/objekt diagrammer
  - Sekvensdiagrammer
- Implementation
  - Kode
- Test.
  - Testplaner
  - Testrapporter

4.1. Iterationsplan

Dette skema viser hvordan vi har opdelt projektperioden.

	Uge 1	Uge 2	Uge 3	Uge 4	Uge 5	Uge 6	Uge 7	Uge 8	Uge 9	Uge 10
Inception										
Elaboration 1										
Elaboration 2										
Elaboration 3										
Elaboration 4										
Rapportskrivning										

Figur 2: Iterationernes fordeling i projektforløbet

Den første periode Inception, bestod hovedsageligt af foranlyse, udarbejdelse af kravsspecifikation, opsætning af udviklingsmiljø og udarbejdelse af rapport-skabelon.
Herefter fulgte 8 uger med udvikling opdelt i 4 Elaborations af 2 uger.
Den sidste uge blev hovedsageligt brugt til rapportskrivning og anden dokumen-tation.

Herunder ses en plan over de punkter der skal være dækket i hver elaboration. Planen er tilpasset løbende gennem projektet, efterhånden som vi har lært mere om de enkelte punkter.
En iteration er først blevet beskrevet i detaljer, umiddelbart før slutningen af den forrige iteration.

4.1.1. Inception

Foranalyse:

- Kan de eksisterede installationspakker bruges, i deres nuværende form?
- Kan installationerne gennemføres, uden at der kræves menneskelig påvirkning (silent install)?
- Hvordan er det muligt at overføre installationspakkerne fra Versions Serveren til Kunde Serveren?
- Hvilket sprog eller framework skal bruges til at udvikle løsningen?
- Hvilket sprog eller framework er bedst egnet til projektet?

Rapport:

- Udarbejd en rapportskabelon, ud fra det format der er påkrævet eksamensprojekter [SKA].

4.1.2. Elaboration 1:

Klient-Server kommunikation:

- Udarbejd server og klient prototyper, der muliggør test.
- Skab forbindelse med WebService fra Opdatering Komponent.
- Lav grundlæggende versionsforespørgsler via WebServicen.
- Undersøg om BITServicen er egnet til overførsel af installationspakkerne.

Opdaterings Komponent:

- Udvikling af Webtop Opdaterings Komponent.
  - Skal kunne finde installerede Versioner af Webtop Produkter.
  - Skal kunne kommunikere med Version Serverens WebService.
  - Skal give Kunden mulighed for at vælge mulige opgraderinger.
  - Skal kunne initialisere download og installation af de valgte opgraderinger.

Version Server:

- Versions Service (WebService)
  - Skal kunne servicere Opdatering Komponenten, med informationer om Versioner.
  - Skal kunne gemme alle Produkt og Versionsoplysninger som XML

Server Installer:

Midlertidig applikation der skal kunne håndtere de overførsler og installationer der igangsættes fra Opdaterings Komponenten.

- Skal implementeres med indbygget simpel webserver.
- Skal have en webapplikation (GUI), der kører på den indbyggede webserver
  - Gør det muligt at følge installationsprocessen fra den klient computer, der startede installationen (via Opdatering Komponenten).

BITS downloader:

Bruges til at overføre installationspakkerne til Kunde Serveren fra Version Serveren.

- Skal implementeres som en assembly, der kan benyttes af Server Installeren.

4.1.3. Elaboration 2:

Version Service:

- Skal kunne håndtere forudsætninger i form af andre versioner.
- Skal kunne logge kundeoplysninger, inkl. Versioner.
- Skal have en Activity log.

Administrations Komponent:

- Det skal være muligt at tilføje, redigere og slette versioner.
- Det skal være muligt at tilføje forudsætninger i form af andre versioner.
- Skal kunne vise oversigt over alle tilføjede Produkter og deres versioner.
- Skal kunne vise de tilgængelige informationer om Kundernes installerede Versioner.

Opdaterings Komponent:

- Skal kunne håndtere versionsforudsætninger i form af andre versioner.
- Kunden skal have mulighed for at opgradere til den nyeste version, han opfylder forudsætningerne for.
- "Vis ikke opdateringer for dette Produkt" skal implementeres.

4.1.4. Elaboration 3:

Opdaterings Service:

Alle applikationer startet af ASPNET brugeren, der er dem Windowsbruger ASP.NET bruger til at eksekvere websites, har af sikkerhedsmæssige årsager, meget begrænsede systemrettigheder.
Installationen af Produkter kræver administrationsrettigheder.
Det ville være meget upraktisk at give ASPNET brugeren administrations-rettigheder, da dette ville være et stort sikkerhedshul og praktisk talt giver samtlige brugere fuld kontrol over serveren.
Derfor er der brug for en WindowsService, der kan køre på Kunde Serveren med disse rettigheder.

Denne service skal herefter overtage Server Installerens opgaver, herudover skal den have følgende funktionaliteter:

- Udføre versions kontrol og opdateringer automatisk, på foruddefineret tidspunkter.
- Servicen skal kunne være i 3 tilstande.
  - Deaktiveret, ingen opgraderinger er mulige.
  - Halvautomatisk, opgraderinger kun mulige gennem Opdaterings Komponenten.
  - Fuldautomatisk, Opdatering Servicen er aktiv og opdatering er også mulig gennem Opdaterings Komponenten.
- Servicen skal være mellemled i kommunikationen mellem Opdaterings Komponenten og Versions Servicen, således at deaktivering er mulig.
- Det skal være muligt at deaktivere muligheden for opgraderinger på hvert enkelt produkt.
- Det skal være muligt at konfigurere servicen via et GUI (AWO-Configurator).

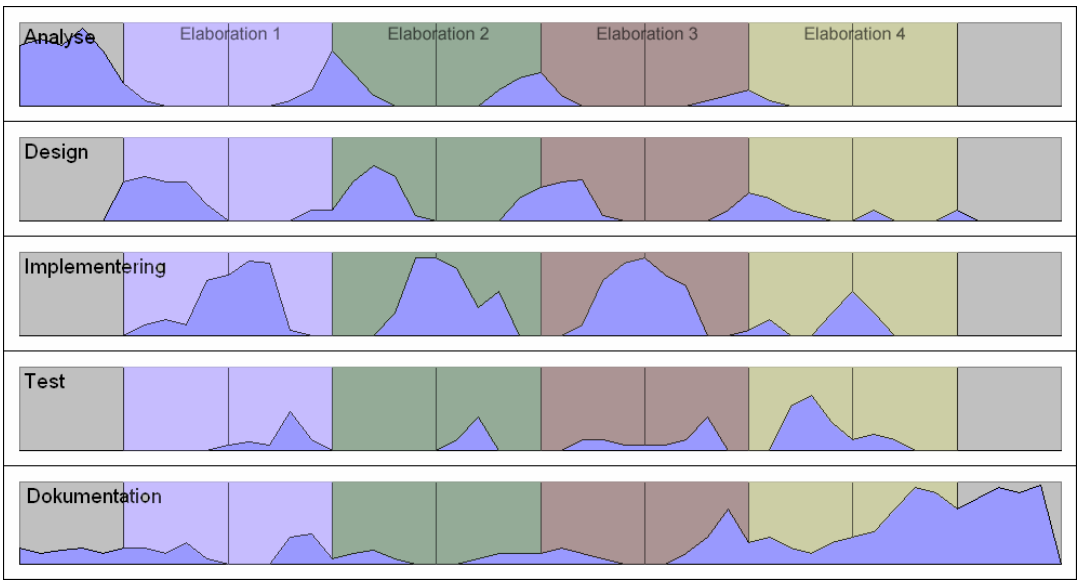
4.1.5. Elaboration 4:

Test:

- Sammensæt en portal baseret på den kommende WebtopONE version 5.0 der skal bruges til test af AWO-Systemet. Portalen skal så vidt muligt ligne og opfører sig som WebtopONE version 5.0 vil. Dette er nødvendigt da den endelige udgave af WebtopONE version 5.0 ikke er færdigudviklet ved projektets afslutning.
- Integration og test af systemet i den specialudviklede portal.
- Test af systemet, med de forskellige dele på hver sin maskine.
- Kontrol af testplaner/rapporter.

4.2. Tidsforbrug

Nedenstående diagram beskriver hvordan tidsforbruget i projektet fordeler sig på de 5 emner: analyse, design, implementering, test og dokumentation.
De lodrette streger angiver at en ny uge begynder og de farvede områder viser de 4 elaborations.
Som man kan se af diagrammet, er der for hver elaboration først analyseret, herefter lavet design, så implementeret og til sidst, udført tests.
Tiden der blev brugt på analyse i skellet mellem 2 elaborations, falder langsomt gennem projektforløbet, efterhånden som løsningen blev fastlagt.
Det samme gælder for tiden til design og implementering, hvorimod tiden til test og dokumentation, stiger mod projektperiodens afslutning.
Der er gennem hele projektforløbet lavet dokumentation i form af artefakter, som use cases, klassediagrammer o.l. Disse er så blevet tilrettet løbende og til sidst finjusteret inden de er blevet sat ind i rapporten.



Figur 3: Fordeling af tidsforbruget i projektforløbet

KAPITEL 5

5. Foranalyse

5.1. Teknologivalg

5.1.1. .NET 2.0

Webtop har besluttet, at deres nye produkter skal udvikles i .NET 2.0 frameworket. Da vores løsning skal passe ind i deres produktprogram, har vi valgt også at bruge denne teknologi.
.NET stiller en række funktionaliteter til rådighed, som vi så vidt muligt har benyttet os af, så som Generics, WebService og WebControls m.m.

5.1.2. XML

I vores løsning var der brug for at overføre informationer via en WebService, som f.eks. Produkt og Versions informationer. Disse informationer skulle også kunne gemmes på harddisken på den centrale Version Server. Derfor har vi valgt at benytte XML som lagringsformat.
XML giver os den fordel at det kan oversættes direkte til en streng (string), som kan overføres problemfrit via WebServicen. Samtidig er der i .NET mulighed for let at gemme XMLen til en tekstfil.
Yderligere er det en fordel at XML-formatet gør det muligt at aflæse data manuelt fra tekstfilerne, da formatet er meget struktureret.Derudover er XML uafhængig af platform og meget fleksibel, når strukturen skal ændres.

5.1.3. WebService

Vi har valgt at bruge en WebService til kommunikationen med den centrale Versions Server. Det giver den fordel, at det er muligt, at forbinde til denne, både fra en webapplikation og en Windows applikation, med den indbyggede funktionalitet i .NET frameworket, hvilket gør det let, senere at skifte dele af løsningen ud. Undervejs fandt vi det f.eks. nødvendigt at skifte, fra at forbinde til servicen direkte fra vores Opdaterings Komponent, til at benytte den WindowsService som også installeres hos Kunden.

5.2. Genbrug af installere

Installere til WebtopONE produkter fra og med version 5.0 bliver udviklet med InstallAware. Resultatet bliver en .msi pakke. Da dette er gældende for alle de produkter vi skal kunne understøtte og der er understøttelse af ”silent install” kan pakkerne bruges direkte.
Hvis det senere besluttes, at der også skal være understøttelse af versioner før 5.0, er det i nogle tilfælde muligt, at ”pakke” ældre installationspakker ind i en .msi pakke og dermed muliggøre opgradering af disse.
Ved at benytte InstallAware opnår vi endnu en fordel, i og med hvert Produkt tildeles et GUID, som registeres i registreringsdatabasen og senere gør det muligt at genkende dem, når de er installeret på Kunde Serveren.

5.3. ”Silent install”

Silent installering vil sige at det ikke er nødvendigt for brugeren at indtaste noget under installationsprocessen, eller andet der kræver ydre påvirkning. I alle de felter der normalt indtastes en værdi, bruges en foruddefineret standardværdi.
Som nævnt ovenfor tillader InstallAware at der udføres ”silent install”.

5.4. Overførsel af installationspakker

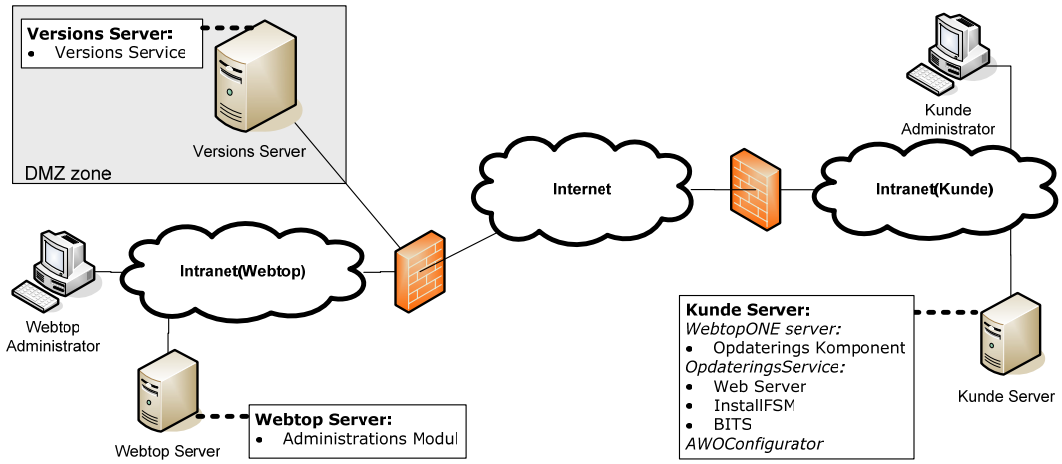
For at muliggøre vores løsning, har vi brug for en metode, til at overføre installationspakkerne fra Versions Serveren til Kunde Serveren. Dette skal kunne

foregå via HTTP protokollen, da det er den eneste protokol/port vi kan være sikker på er tilgængelig hos alle Kunderne.
BITServicen fra Microsoft er meget veldokumenteret og udviklet til formål som vores. Derfor har vi valgt at basere produktet på denne. BITS-servicen, og anvendelsen af denne, er nærmere beskrevet under afsnittet 7.4.3 Opdaterings Service

KAPITEL 6

6. Analyse

I analysenfasen af hver iteration har vi behandlet de delkomponenter, der skulle udvikles eller revideres i iterationen og hvordan denne skulle indpasses i det samlede system, der er illustreret herunder.



Figur 4: Oversigt over systemet

Kunde Serveren er som navnet angiver, placeret hos Kunden eller hosted for dem.
Webtop Serveren er placeret hos Webtop, men kunne ligeledes være hosted.
Version Serveren vil være placeret i Webtops DMZ-zone.
I boksene er beskrevet hvilke delelementer der er på hvilke servere.

6.1. Beskrivelse af delelementer

Herunder er en kort beskrivelse af hvilke områder de forskellige elementer har ansvar for.

6.1.1. Kunde Server

WebtopONE Server
På Kunde Serveren er der i forvejen en WebtopONE installation. I det komponent katalog der er tilknyttet installationen, skal Opdaterings Komponenten placeres, således at Kunde Administratoren kan sætte den ind på sin portal.

Opdatering Komponent:
Opdatering Komponenten er Kunde Administratorens interface til halvautomatiske opgraderinger af hans WebtopONE produkter.

Opdaterings Service
Opdaterings Servicen er ansvarlig for kommunikationen til Versions Serveren.
Den skal være bindeled mellem Opdatering Komponenten og Versions Serveren.
Den skal også stå for de fuldautomatiske skedulerede opdateringer.
Opdaterings Servicen består af flere delelementer, der hver er ansvarlig for sit område.

Webserver
Webserveren har til formål at præsentere Kunde Administratoren for et vindue, der gør det muligt at følge opdateringsforløbet, da IIS'en i visse tilfælde skal genstartes i forbindelse med en opdatering.

InstallFSM
InstallFSM er en finite state machine (FSM), der skal holde styr på hvad der skal udføres i de forskellige trin i opdateringsprocessen. Samtidig skal den kunne videregive informationer om hvilket stadie, opdateringen er i.

BITS
BITS er en WindowsService der kan bruges til at overføre data binært over en HTTP forbindelse.
I vores løsning skal BITS bruges til at overføre installationspakkerne.

AWOConfigurator
Formålet med AWOConfigurator at er gøre det muligt for Kunden, at ændre instillingerne for Opdaterings Servicen.

6.1.2. Versions Server

Versions Services
Versions Servicen skal gøre det muligt at vedligeholde Versions Serveren fra Administrations Komponenten, samt gøre det muligt for Kunden, at hente opgraderinger.

6.1.3. Webtop Server

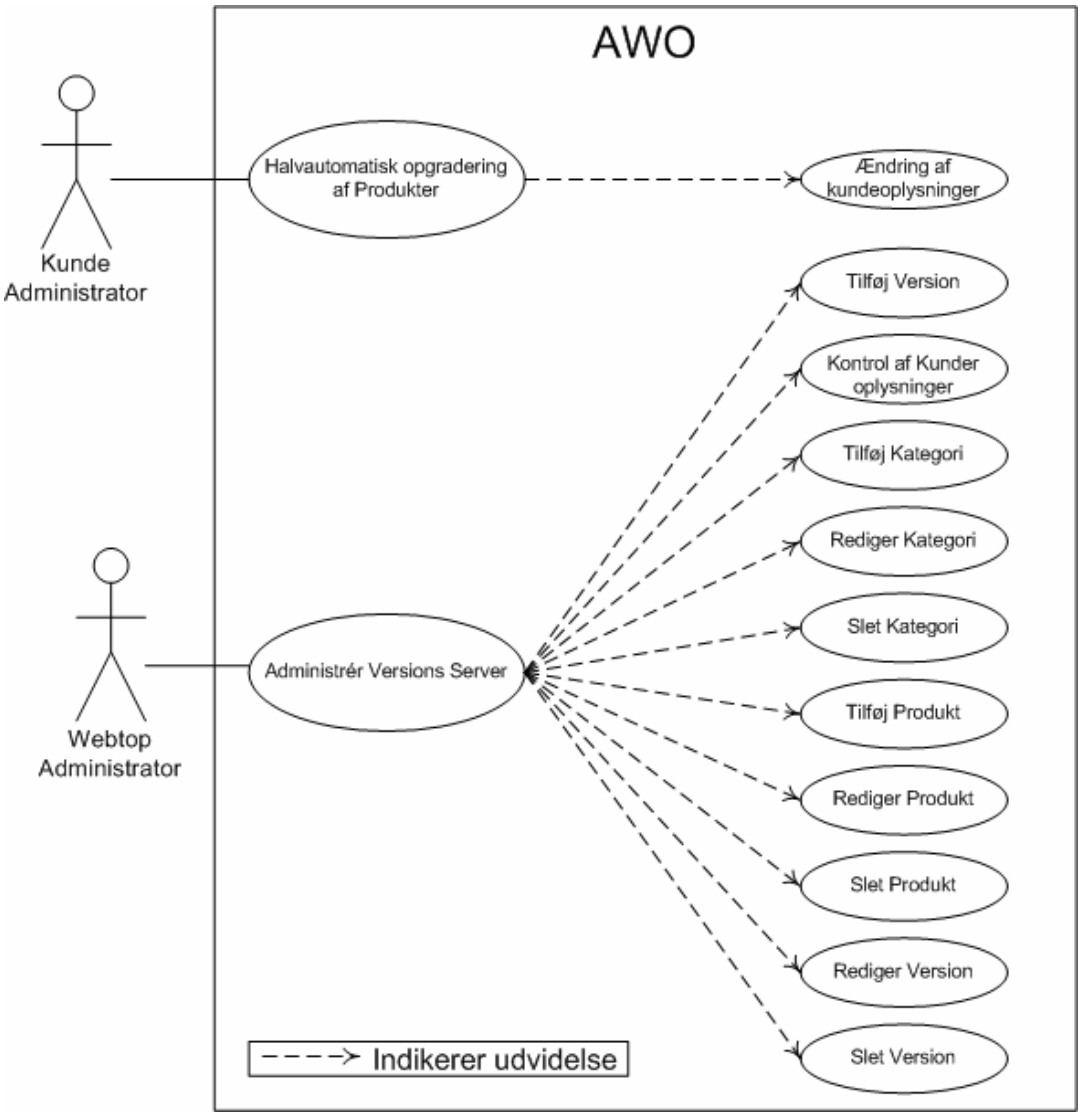
Webtop Serveren er den WebtopONE server, hvor Webtop har sin portal. Det er her Administrations Komponenten skal være tilgængelig.

Administrations Komponent
Administrations Komponenten skal gøre det muligt for Webtops personale, at vedligeholde Versions Serveren. Den kommunikerer med Versions Servicen, når der skal vises informationer og laves ændringer.

6.2. Overordnet use cases

De overordnede use cases er skrevet set fra brugerens synspunkt, dvs. Kunde eller Webtop Administratoren. Dette betyder at alle tekniske detaljer er udeladt.
Disse overordnede use cases er nærmere beskrevet i Design afsnittet, hvor tekniske detaljer, så som kommunikation o.l. er medtaget.

6.2.1. Use Case Diagram



Figur 5: Use Case Diagram

6.2.2. Overordnet use case I – Halvautomatisk Opgradering

Følgende use case beskriver hvordan en Kunde kan opgradere sine Produkter halvautomatisk ved hjælp af Opdatering Komponenten.

Primær aktør: Kunde Administrator.

USE CASE I	Halvautomatisk Opgradering
Forudsætninger	• Der kan skabes forbindelse mellem Opdatering Komponenten og Versions Serveren.
Succes kriterier	• Produkterne er opdaterede til de ønskede Versioner.
Hoved forløb	
1	Kunden logger sig på sin egen portal med administrator rettigheder.
2	Kunden præsenteres for de mulige versionsopdateringer.
3	Kunden vælger de Produkter han ønsker at opdatere.
4	Opgraderingerne installeres. Kunden præsenteres for et statusvindue, hvor han kan følge installationsprocessen.
5	Kunden præsenteres for installations resultatet.
Udvidelser	
1a	Kunden har ikke administratorrettigheder. 1. Opdatering Komponenten kan ikke vises. Dette håndteres af WebtopONE. 2. Use casen afsluttes.
2a	Der er ingen tilgængelige opdateringer. 1. Kunden informeres herom. Use casen afsluttes.
2b	Kunden ønsker ingen af de, tilgængelige opdateringer. 1. Use casen afsluttes.
2c	De påkrævede Kundeoplysninger er ikke registreret: 1. Kunden præsenteres for en formular hvor informationerne kan indtastes.
2d	Hvis kunden ønsker at ændre sine oplysninger, kan han via et link, blive præsenteret for en formular, der gør dette muligt. (Use Case Ia)

Use case Ia er en udvidelse til use case I. Den beskriver hvordan Kunden kan opdatere sine oplysninger via Opdatering Komponenten.

USE CASE Ia	Ændring af kundeoplysninger
Forudsætninger	• Use case I er udført til punkt 2
Succes kriterier	• Kundens oplysninger er registreret.
Hoved forløb	
1	Kunden opdaterer sine oplysninger.
2	Oplysningerne gemmes.
3	Opdatering Komponenten bekræfter at ændringerne er registreret.

6.2.3. Overordnet use case II – Administrér Versions Server

Følgende use case beskriver hvordan en Webtop Administrator kan vedligeholde Versions Serveren. Dette gøres gennem Administrations Komponenten.
Primær aktør: Webtop Administrator.

USE CASE II	Administrér Versions Server
Forudsætninger	• Administratoren er identificeret og bekræftet. • Der kan skabes forbindelse mellem Administrations Komponenten og Versions Serveren.
Succes kriterier	• De ønskede oplysninger er kendt og de ønskede ændringer er blevet registreret.
Hoved forløb	
1	Administratoren logger på sin portal.
2	Administratoren finder de ønskede oplysninger og foretager de ønskede ændringer.
Udvidelser	
2a	Kontrol af Kunder oplysninger (Use Case IIa)
2b	Tilføj Kategori (Use Case IIb)
2c	Rediger Kategori (Use Case IIc)
2d	Slet Kategori (Use Case IId)
2e	Tilføj Produkt (Use Case IIe)
2f	Rediger Produkt (Use Case IIf)
2g	Slet Produkt (Use Case IIg)
2h	Tilføj Version (Use Case IIh)
2i	Rediger Version (Use Case IIi)
2j	Slet Version (Use Case IIj)

Use case IIa beskriver hvordan Administratoren kan kontrollere en Kundes oplysninger.

USE CASE IIa	Kontrol af Kunder oplysninger
Forudsætninger	• Use case II er udført til punkt 2.
Succes kriterier	• Kundens oplysninger er kendt af Administratoren.
Hoved forløb	
1	Administratoren vælger den Kunde der ønskes oplysninger fra.
2	Versions Serveren returnerer de ønskede oplysninger, som præsenteres for Administratoren.

Use case IIb-IId beskriver administrationen af Kategorier.

USE CASE IIb	Tilføj ny Kategori
Forudsætninger	• Use case II er udført til punkt 2.
Succes kriterier	• Den nye Kategori er lagt ind i systemet.
Hoved forløb	
1	Administratoren udfylder informationer om Kategorien.
2	Administrations Komponenten bekræfter at Kategorien er lagt ind.
Udvidelser	
2a	En Kategori med det indtastede navn findes allerede. 1. Registreringen fejler og Administratoren informeres om problemet. Use casen fortsættes fra step 1, med de indtastede værdier bevaret.

USE CASE IIc	Rediger Kategori
Forudsætninger	• Use case II er udført til punkt 2.
Succes kriterier	• Den valgte Kategori er opdateret.
Hoved forløb	
1	Administratoren vælger en Kategori.
2	Administratoren opdaterer informationer om Kategorien.
3	Administrations Komponenten bekræfter at Kategorien er opdateret.
Udvidelser	
2a	En Kategori med det indtastede navn findes allerede. 1. Registreringen fejler og Administratoren informeres om problemet. Use casen fortsættes fra step 1, med de indtastede værdier bevaret.

USE CASE IId	Slet Kategori
Forudsætninger	• Use case II er udført til punkt 2.
Succes kriterier	• Den valgte Kategori er opdateret.
Hoved forløb	
1	Administratoren vælger en Kategori der skal slettes.
2	Administratoren bekræfter sletningen.
3	Administrations Komponenten bekræfter at Kategorien er opdateret.
Udvidelser	
2a	Administratoren fortryder sletningen. 1. Use casen afsluttes.

Use case IIe-IIg beskriver administrationen af Produkter.

USE CASE IIe	Tilføj ny Produkt
Forudsætninger	• Use case II er udført til punkt 2.
Succes kriterier	• Det nye Produkt er lagt ind i systemet.
Hoved forløb	
1	Administratoren udfylder informationer om Produktet.
2	Administratoren tilføjer versioner.
3	Administrations Komponenten bekræfter at Produktet er lagt ind.
Udvidelser	
3a	Et Produkt med det indtastede navn findes allerede. 1. Registreringen fejler og Administratoren informeres om problemet. Use casen fortsættes fra step 1, med de indtastede værdier bevaret.

USE CASE IIf	Rediger Produkt
Forudsætninger	• Use case II er udført til punkt 2.
Succes kriterier	• Det valgte Produkt er opdateret.
Hoved forløb	
1	Administratoren vælger et Produkt.
2	Administratoren opdaterer informationer om Produktet.
3	Administratoren opdaterer Produktets Versioner (Use case IIh-IIj).
4	Administrations Komponenten bekræfter at Produktet er opdateret.
Udvidelser	
4a	Et Produkt med det indtastede navn findes allerede. 1. Opdateringen fejler og Administratoren informeres om problemet. Use casen fortsættes fra step 1, med de indtastede værdier bevaret.

USE CASE IIg	Slet Produkt
Forudsætninger	• Use case II er udført til punkt 2.
Succes kriterier	• Det valgte Produkt er slettet.
Hoved forløb	
1	Administratoren vælger Produktet der skal slettes.
2	Administratoren bekræfter sletningen.
3	Administrations Komponenten bekræfter at Produktet er opdateret.
Udvidelser	
2a	Administratoren fortryder sletningen. 1. Use casen afsluttes.

Use case IIh-IIj beskriver administrationen af Versioner.

USE CASE Iih	Tilføj ny Version
Forudsætninger	• Use case II er udført til punkt 2.
Succes kriterier	• Den nye Version er lagt ind i systemet.
Hoved forløb	
1	Administratoren vælger det Produkt der skal have tilføjet en ny Version.
2	Administratoren udfylder informationer, så som versions nummer, ændringshistorik m.m.
3	Administratoren vælger den installationspakke der skal tilknyttes Versionen
4	Administrations Komponenten bekræfter at versionen er lagt ind.
Udvidelser	
5a	Versionsnummeret er ugyldigt, eller findes allerede for dette Produkt. 2. Registreringen fejler og Administratoren informeres om problemet. Use casen fortsættes fra step 2, med de indtastede værdier bevaret.

USE CASE Iii	Rediger Version
Forudsætninger	• Use case II er udført til punkt 2.
Succes kriterier	• Versionen er opdateret.
Hoved forløb	
1	Administratoren vælger det Produktet, Versionen er tilknyttet.
2	Administratoren vælger Versionen der skal opdateres.
3	Administratoren redigere de ønskede oplysninger og vælger evt. ny installationspakke.
4	Administrations Komponenten bekræfter at versionen er opdateret.
Udvidelser	
4a	Versionsnummeret er ugyldigt, eller findes allerede for dette Produkt. 3. Registreringen fejler og Administratoren informeres om problemet. Use casen fortsættes fra step 3, med de indtastede værdier bevaret.

USE CASE IIj	Slet Version
Forudsætninger	• Use case II er udført til punkt 2.
Succes kriterier	• Versionen er slettet.
Hoved forløb	
1	Administratoren vælger det Produktet, Versionen er tilknyttet.
2	Administratoren vælger de Versionen der skal slettes.
3	Administratoren bekræfter sletningen.
4	Administrations Komponenten bekræfter at versionen er slettet.
Udvidelser	
3a	Administratoren fortryder sletningen. 4. Use casen afsluttes.

KAPITEL 7

7. Design og implementering

Da dette projekt er udviklet efter Unified Process metoden har vi arbejdet iterativt og derfor er designet og koden også udviklet iterativt.
Vi har efter hver iteration haft et fungerende produkt, der så er blevet videreudviklet i den efterfølgende iteration. En iteration bestod af helt nye artefakter eller omskrivning og forbedring af eksisterende.
Det, at der fandtes en fungerende løsning, som minimum efter hver iteration, gjorde det muligt at vurdere om de ideer vi havde, fungerede i praksis. Samtidig blev det lettere for Webtop, at følge udviklingen af projektet.
Yderligere er debugmulighederne større, når koden kan eksekveres.

For at have et produkt, der så tidligt som muligt, repræsenterede den endelige løsning, valgte vi ofte at bruge forsimplinger i de tidlige iterationer. Som eksempel kan nævnes, at der var direkte kommunikation fra Opdaterings Komponenten til WebServicen på Versions Serveren. Dette gjorde at vi kunne teste mange dele af løsningen, uden at skulle arbejde med en WindowsService, der har meget dårlige debugmuligheder. Senere ændrede vi kommunikationsvejen således, at Opdaterings Servicen, der skulle stå for de fuldautomatiske opdateringer også fungerede som mellemled i den førnævnte kommunikation.

Under udviklingen har vi benyttet Microsoft Visual SourceSafe til at styre adgangskontrollen til artefakterne. Dette er en stor fordel når man er flere der arbejder med samme løsning, idet at kun en person kan arbejde med de enkelte arfakter ad gangen. Samtidig giver det mulighed for at se tidligere versioner af filerne, under forudsætning af at disse er blevet checket ind løbende.

De følgende afsnit beskriver de overvejelser, vi har gjort os i forbindelse med design og implementering af løsningen.

7.1. Teknologianvendelse

I projektet har vi benyttet forskellige teknologier der var tilgængelige i .Net frameworket. De vigtigste af disse er beskrevet i dette afsnit.

7.1.1. Generics i .NET

I .NET 2.0 er der kommet en række såkaldte generics, heriblandt *List<T>*.
Generics konceptet går kort fortalt ud på, at man kan erklære en generic med en vilkårlig type, hvorefter den vil være bundet til denne type, de er med andre ord *typestærke* efter oprettelse.
Det kan sammenlignes med et array, hvor hvis man opretter et string-array kan man kun "putte" stings i det.
Et eksempel på en generic, som vi har benyttet meget i dette projekt er *List*. *List* minder meget om et array, men har bl.a. de fordel at den kan gennemløbes med f.eks. en foreach statement og at den fungere som en collection, hvilket er en stor fordel når elementer ofte skal tilføjes og fjernes fra *Listen*.
Det er også muligt at arve fra generics, hvilket vi bl.a. har benyttet i *CategoryCollection*, der arver fra *List<Category>*.

7.1.2. GUID

Da vi har flere elementer vi skal kunne kende fra hinanden, så som Produkter, havde vi brug for et unik ID. Man kunne have benyttet fortløbende numre, men dette kan give problemer, efter sletning o.l.
Da vi allerede havde et GUID (**G**lobally **U**nique **Id**entifier) fra installations pakkerne og GUIDs er understøttet af .NET frameworket, var dette et åbenlyst valg. Vi skulle under alle omstændigheder bruge Produktets GUID til at genkende installerede Produkter på Kundens WebtopONE server.

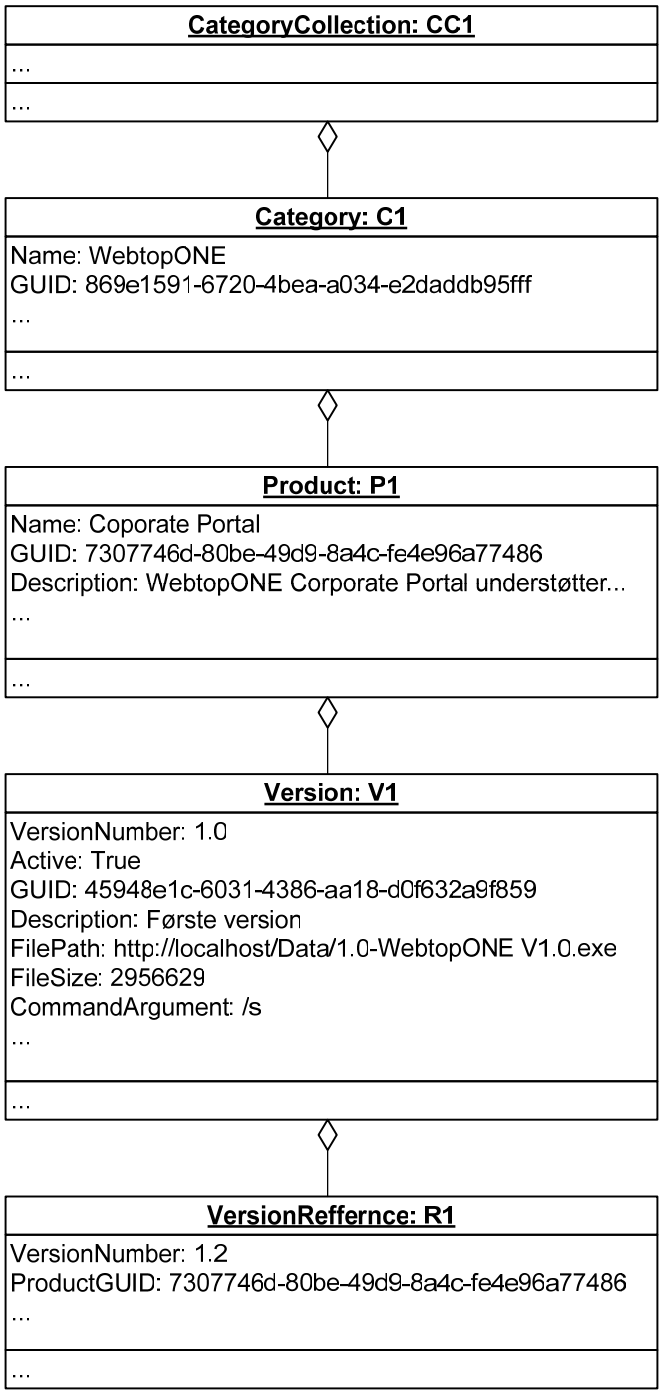
7.1.3. IXmlObj Interfacet

Som lagringsformat for både Produkt/Versions- og Kunde oplysninger, har vi valgt XML.
I de tilfælde hvor objekter skal kunne lagres, har vi valgt at lade vores objekter og xml-repræsentationen være så lig hinanden som muligt.
Alle objekter der skal kunne lagres som XML implementerer et interface, *IXmlObj*, der kræver at de kan danne xmlnode-repræsentationer af sig selv og at de kan genskabe sig fra xmlnoder igen.

```
public interface IXmlObj
{
    void FromXml(XmlNode Element);
    XmlNode ToXml(XmlDocument XD);
}
```

Det skal dog bemærkes at det ikke er al XML der bliver genereret, der har en direkte objekt represæntation. Den XML som kun fungerer som log, er et eksempel på XML unden tilsvarende objekter.

I de tilfælde hvor der er tale om en hierarkisk struktur, som med Kategori, Produkt og Version, er det stadig hver enkelt del, der er ansvarlig for sin egen transformation.
Kategori benytter sin egen *ToXml()* som kalder Produkternes *ToXml()* når disse skal repræsenteres. Ligeledes benytter hvert Produkt Versionernes *ToXml ()* til at generere xmlnoder for sine Versioner.
Samme metode benyttes når objekterne skal genskabes vha. *FromXml()*.
Denne opbygning giver os den fordel, at ændringer langt nede i hierarkiet, kun skal ændres et sted, for at blive afspejlet i hele systemet. Samtidig gør det det lettere at skifte dele af objekt-strukturen ud.
De næste sider viser et eksempel på et sådan hierarki i form af objekt/xml stuktuten for produkter.



Figur 6: Objektstruktur for Kategorier, Produkter og Versioner

```
<ProductCategories>
  <ProductCategory>
    <Name>WebtopONE</Name>
    <GUID>869e1591-6720-4bea-a034-e2daddb95fff</GUID>
    <Products>
      <Product>
        <Name>Corporate Portal</Name>
        <GUID>7307746d-80be-49d9-8a4c-fe4e96a77486</GUID>
        <Description>
          WebtopONE Corporate Portal understøtter...
        </Description>
        <Versions>
          <Version>
            <VersionNumber>1.0</VersionNumber>
            <Active>True</Active>
            <GUID>45948e1c-6031-4386-aa18-d0f632a9f859</GUID>
            <Description>Første version</Description>
            <FilePath>
             http://localhost/Data/1.0-WebtopONE V1.0.exe
            </FilePath>
            <FileSize>2956629</FileSize>
            <CommandArgument>/s</CommandArgument>
            <RequiredVersions>
              <VersionReference>
                <VersionNumber>1.2</VersionNumber>
                <ProductGUID>
                  7307746d-80be-49d9-8a4c-fe4e96a77486
                </ProductGUID>
              </VersionReference>
              ...
            </RequiredVersions>
          </Version>
          ...
        </Versions>
      </Product>
      ...
    </Products>
  </ProductCategory>
  ...
</ProductCategories>
```

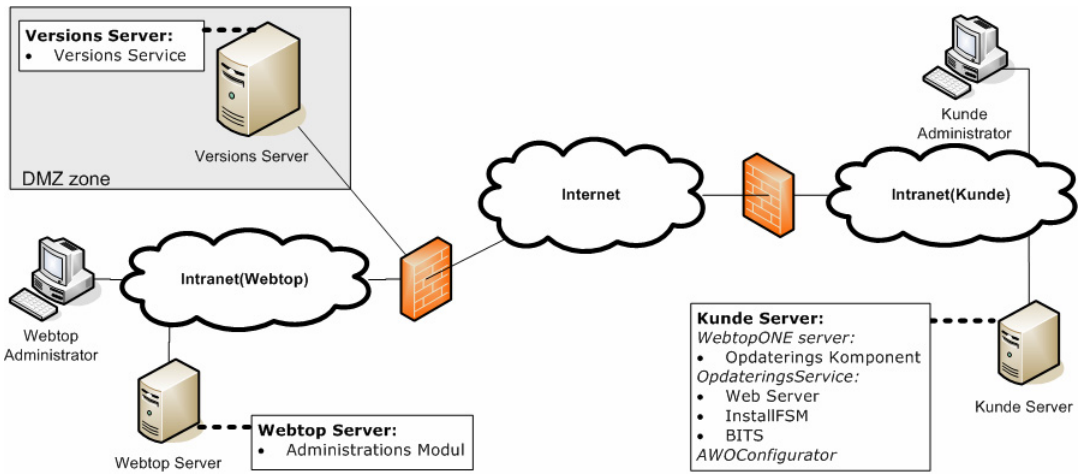
Figur 7: XML representation af strukturen for Kategorier, Produkter og Versioner

7.2. Det samlede system

Dette afsnit beskriver den samlede løsning, samt de dele der har indflydelse på det.

7.2.1. Systemoversigt

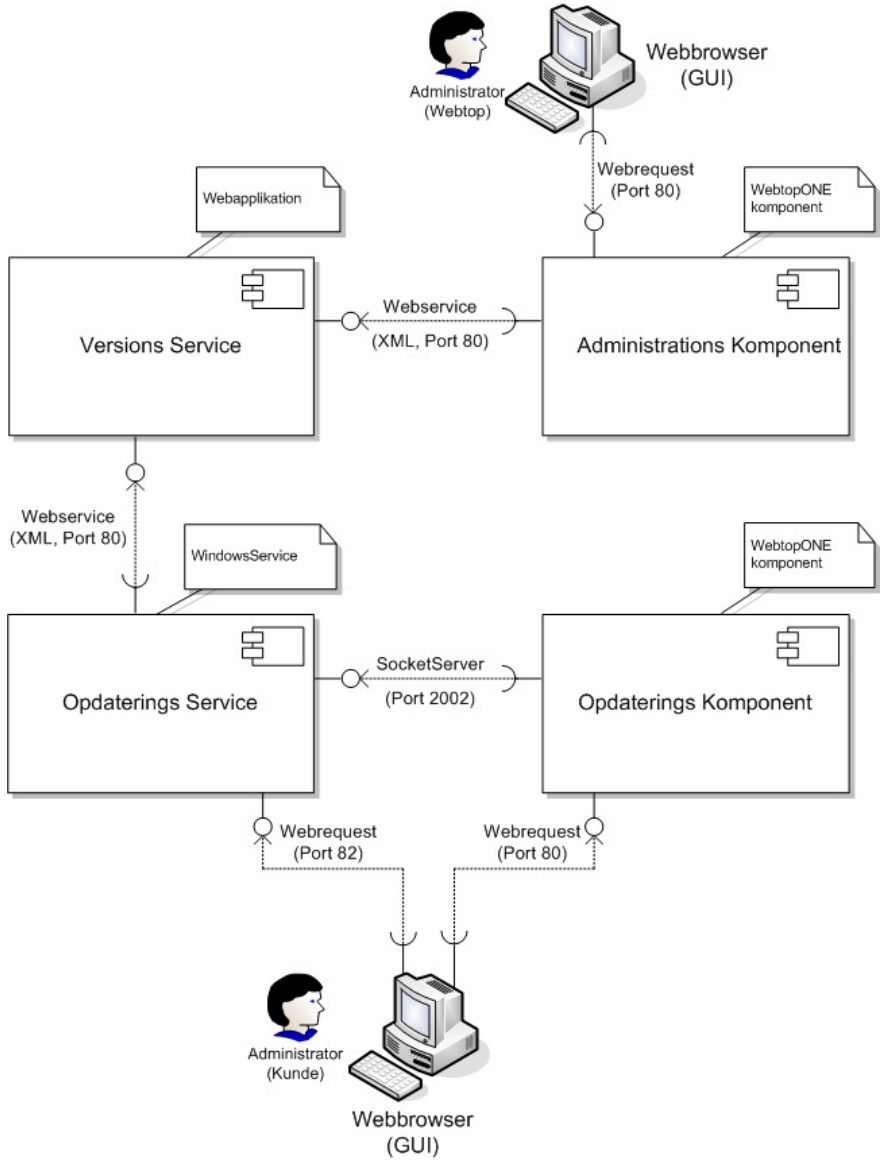
Som beskrevet i Analyse afsnittet, er der 3 servermaskiner involveret i vores løsning, Kunde Serveren, Webtop Serveren og Versions Serveren.
De 3 servere er ansvarlige for hver deres del af løsningen, fordelt på en række delelementer, realiseret i form af .NET projekter.



Figur 8: Oversigt over systemet

7.2.2. Kommunikationsoversigt

Som beskrevet er løsningen realiseret som en række .Net projekter. Nedenstående diagram viser hvordan kommunikationen mellem de 4 hovedelementer og brugerne foregår.



Figur 9: Oversigt af kommunikationen mellem delkomponenterne i systemet

7.2.3. Opgraderings logik

Dette afsnit beskriver hvordan systemet afgør hvilken Version af et Produkt, der skal præsenteres for Kunden som det nyeste.

I de fleste tilfælde er opgraderings logikken meget simpel; den nyest tilgængelige version, er den version med det højeste versionsnummer.
Der er dog 2 ting der kan påvirke dette:

1. En opgradering kan være afhængig af at et andet Produkt har en hvis minimums version.
2. En opgradering kan også være afhængig af, at den installerede version af Produktet selv har en vis version. Dette benyttes i de tilfælde hvor en version kræver ”håndinstallation”. Det vil typisk være fordi opgraderingsprocessen er meget individuel og derfor er svær at automatisere. En sådan ”håndinstallationsversion” vil blive oprettet på Versions Serveren uden installations fil og vil blive markeret inaktiv.

Systemet vil arbejde sig baglæns gennem versionerne, indtil den finder en version der er version uden uopfyldte forudsætninger, eller versionen ikke længere er nyere end den installerede.
Denne struktur tillader at der kan frigives versioner, der retter fejl, selv om der eksisterer en nyere ”håndinstallation” på Versions Serveren.

I de tilfælde hvor der er nye aktive versioner, men der ikke er nogen, hvor alle forudsætningerne er opfyldt, vil Opdaterings Komponenten vise den nyeste af disse versioner med en forklaring på, hvilke forudsætninger der ikke er opfyldt. Det vil ikke være muligt at opgradere til en sådan version, før forudsætningerne bliver opfyldt.

Opgraderings Servicen vil ignorere alle versioner der har uopfyldte krav, når den foretager den fuldautomatiske opgradering. Servicen gentager automatisk installationsprocessen så længe der er opgraderinger uden uopfyldte krav.

For at illustrerer hvordan opdaterings logikken fungerer har vi konstrueret et scenarie, der dækker de interessante tilfælde.

Versioner på Version Serveren	
Version	*Bemærkning*
WebtopONE	
4.6	Deaktiveret version
4.6.1	Ingen
5.0	Deaktivet version
5.1	Kræver WebtopONE 5.0
NoticeBoard	
4.6	Deaktiveret version
4.6.1	Kræver WebtopONE 4.6.1
5.0	Kræver WebtopONE 5.0

Installeret hos kunden:

- WebtopONE 4.6
- Noticeboard 4.6

Kunden åbner Opgraderings Komponenten for at opgradere til de nyeste versioner.

Den nyeste WebtopONE version er 5.1, men den er afhængig af den deaktiverede version 5.0. Da Kunden kun har version 4.6, altså mindre end 5.0 kan der ikke opgraderes til denne version.
Den næste version i rækken er version 5.0, men da dette er en deaktiveret version, kan denne heller ikke installeres.
Derfor er der kun version 4.6.1 tilbage, da denne version ikke har nogen krav, vil det være muligt at opgradere til den og denne opgradering vises i komponenten.

Nyeste Noticeboard er 5.0, men denne er afhængig af at WebtopONE 5.0 eller nyere er installeret.
Version 4.6.1 af Noticeboard kræver at version 4.6.1 eller nyere af WebtopONE er installeret, derfor kan denne version ikke installeres.
Den næste version er 4.6, men da denne version ikke er nyere end den installerede, er den ikke interessant.
Der er derfor ikke nogle opgraderinger tilgængelige. For at gøre kunden opmærksom på at der er nyere versioner tilgængelige vises version 5.0 dog alligevel i komponenten, uden det er muligt at opgradere Produktet. Der er også en

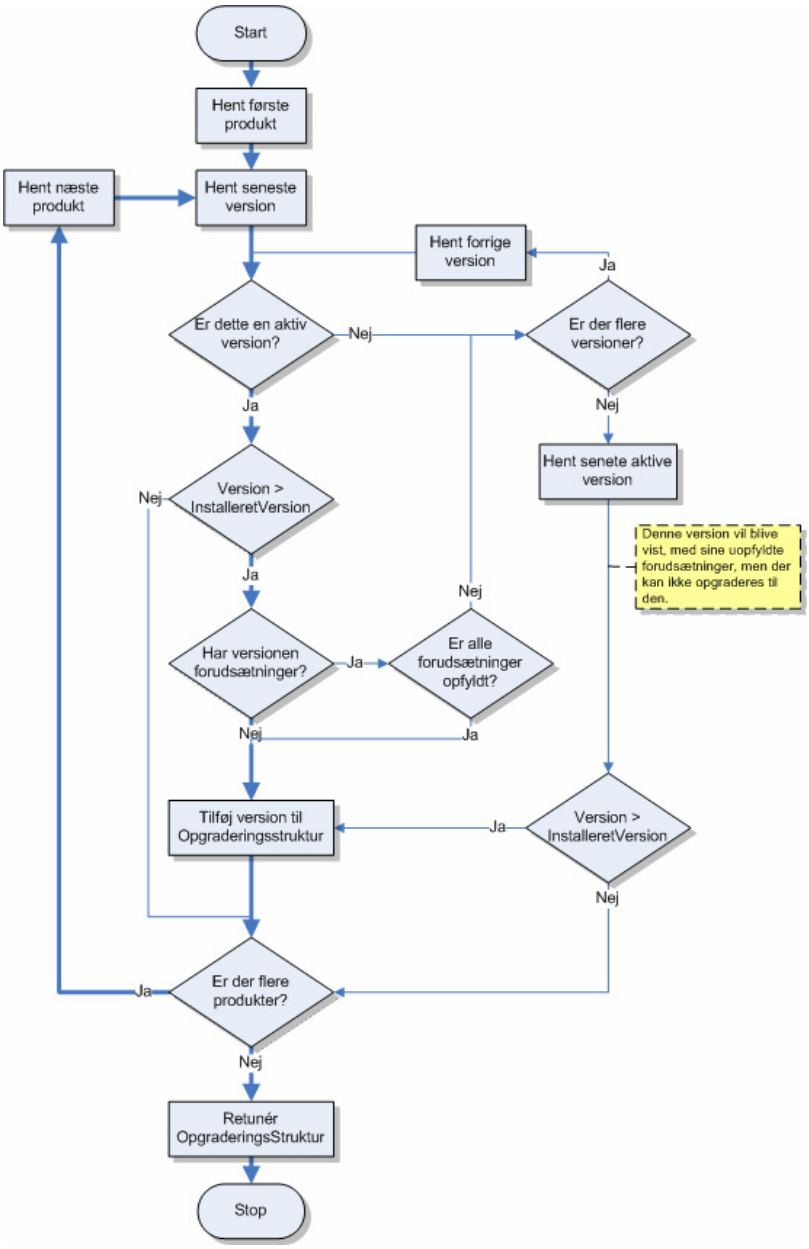
liste over de Produktversioner der er skyld i at opgraderingen ikke er mulig, altså WebtopONE 5.0.

Efter opgraderingen har kunden nu WebtopONE 4.6.1, når komponenten åbnes igen vil der kun være version 5.1 tilgængelige. Komponenten vil nu gøre kunden opmærksom på, at der er nyere versioner, der dog kræver at han får foretaget en manuel installation.
Noticeboard 4.6.1 har nu opfyldt sine krav og Kunden kan opgradere den.

Efter Noticeboard er opgraderet vil både WebtopONE og Noticeboard vises med en bemærkning om, at de er afhængige af en manuel installation af WebtopOne 5.0.

7.2.4. Flow for installations logik

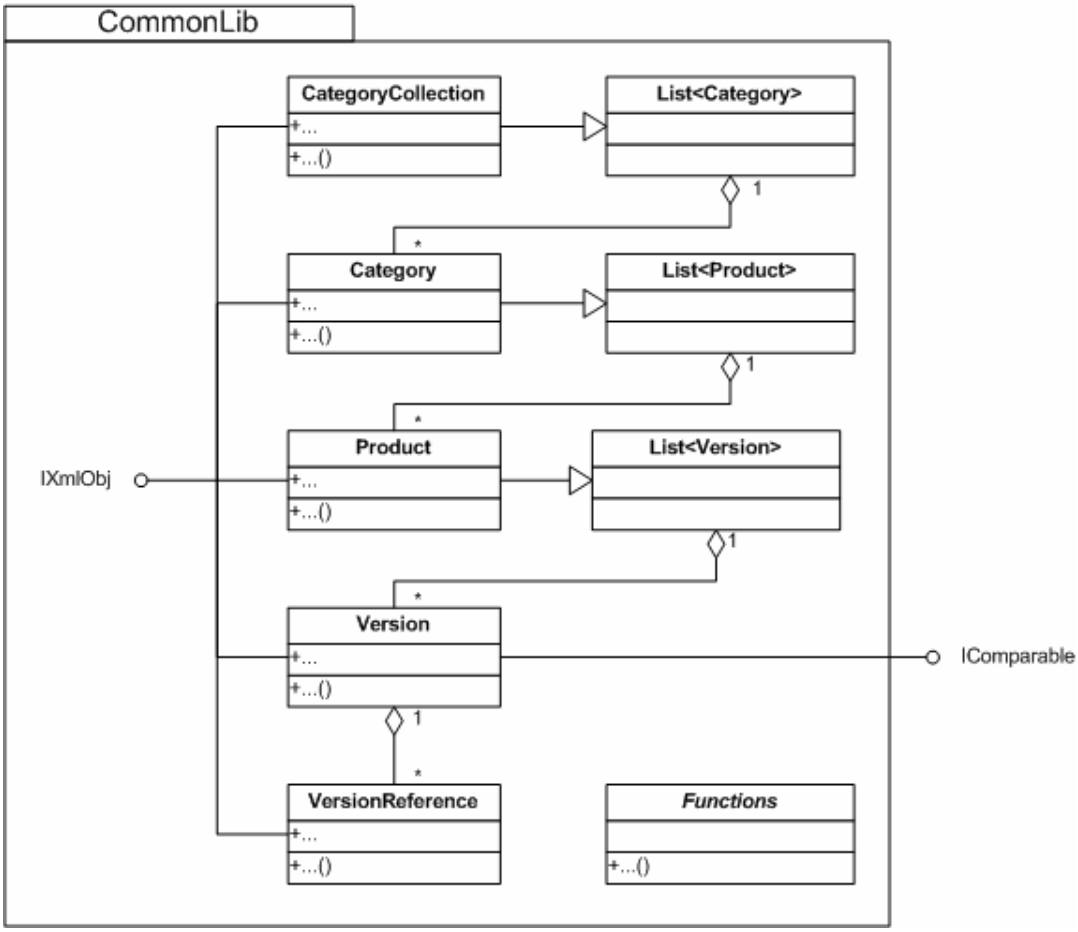
Flowet for hvilke opgraderinger der præsenteres er opsummeret i nedenstående diagram.



Figur 10: Installationslogik

7.2.5. CommonLib

CommonLib bliver brugt af flere af projekterne. Den vigtigste del er Produkt strukturen, der i vid udstrækning baserer sig på generic'en List.
Derudover er der en abstrakt klasse, *Functions*, der indholder en række funktioner der er nyttige i flere sammenhænge.



Figur 11: CommonLib klassebiblioteket

7.3. Versions Server

Versions Serveren er kernen i vores løsning. Det er den der holder styr på hvilke Produkter der er tilgængelige og i hvilke versioner. Det er samtidig den der holder styr på hvilke Produkter Kunderne har installeret og hvilke opgraderinger de har påbegyndt og hvilke der er gennemført.
Serveren vil som udgangspunkt være en maskine placeret hos Webtop, men den kunne i princippet være placeret et vilkårligt sted.
Adgangen til Versions Serveren foregår, både for Kunderne og Administratorerne, via en WebService, der er beskrevet herunder.

7.3.1. Versions Service (WebService)

På Versions Serveren kører en WebService kaldet Versions Service, der har til ansvar, at servicere de øvrige delsystemer.
Fra kundesiden er det en WindowsService, Opdaterings Servicen, der kommunikerer med WebServicen. Se afsnit 7.4.3.
Fra Webtop er det Administrations Komponenten der kobler op mod Versions Servicen. Se afsnit 7.6

Ved første kald til Versions Servicen hentes oplysninger om samtlige Produkter og deres Versioner fra Products.xml filen der er gemt på Versions Serveren. Ud fra disse oplysninger skabes hele objekt hierarkiet med Kategorier, Produkter og Versioner, som beskrevet i afsnit 7.2.5. Denne objektstruktur bliver herefter gemt i en Application variabel, der er delt for alle instancer af Versions Servicen. Dette er gjort for at forbedre performance og undgå race-conditions ved læsning og skrivning til filen.
Når Opdaterings Servicen eller Administrations Komponenten herefter har brug for Produktoplysninger, bliver de hentet fra denne Application-variabel.
Når en Webtop Administrator ændre Produktoplysninger via Administrations Komponenten, vil Versions Servicen gemme disse både i Application-variablen og til Products.xml filen, således at disse altid er i sync.

For at gøre det muligt for en Webtop Administrator at se hvilke Produkter en Kunde har, f.eks. i forbindelse med en fejl ved en opdatering, gemmer systemet disse oplysninger i XML filer. Der er en oversigt der indeholder oplysninger for

alle kunder Customer.xml, der opdateres hver gang der er sket ændringer. Derudover har hver Kunde en XML fil, der indeholder en activitylog. Hver entry i activityloggen består af Kundens Versionsoplysninger, samt hvad der forsøges opdateret. Efter processen er afsluttet, tilføjes entryen Kundens nye versionsoplysninger.

Vi har valgt at lade al kommunikation foregå primitive typer og arrays af primitive typer. Dette gør at vi undgår problemer med at WebServicen ikke kan håndtere komplekse objekter. I de tilfælde hvor vi har brug for at repræsentere komplekse strukturer, har vi benyttet en xml streng. Se mere i afsnittet Teknologivalg, afsnit 5.1.

7.3.2. Sikkerhed

Versions servicen er en WebService og har dermed som standard ikke nogen sikkerhedsforanstaltninger. Derfor er det muligt, hvis man kalder de rigtige WebMethods at aflæse og ændre oplysningerne på Versions Serveren. Dette er naturligvis en sikkerhedsbrist, der skal rettes op på, før Systemet kan bruges forsvarligt.
I den nuværende version af løsningen, er der lavet en meget simpel form for kontrol, for at sikre at det kun er Webtop Administratorer der kan foretage ændringer. I de funktioner der gør det muligt at ændre produktoplysningerne checkes det om IP-adressen kan verificeres. De tilladte IP-adresser specificeres i web.config.
Dette giver dog kun en begrænset sikkerhed. På længere sigt skal identiteten bekræftes mere effektivt f.eks. ved brug af digitale signaturer.

7.4. Kunde Server

Kunde Serveren er den server Kunden har sin WebtopONE installationer på.
Det er på denne maskine alle opgraderinger skal installeres på via vores løsning. Det er derfor også her Opdaterings Servicen og Opdaterings Komponenten skal installeres.

7.4.1. Use case 1 – Halvautomatisk Opgradering

Nedenstående use case er en teknisk uddybelse af overordnet use case I fra analyse afsnittet. Den beskriver forløbet for en Kunde Administrator der vil opgradere sine Produkter via Opdaterings Komponenten.

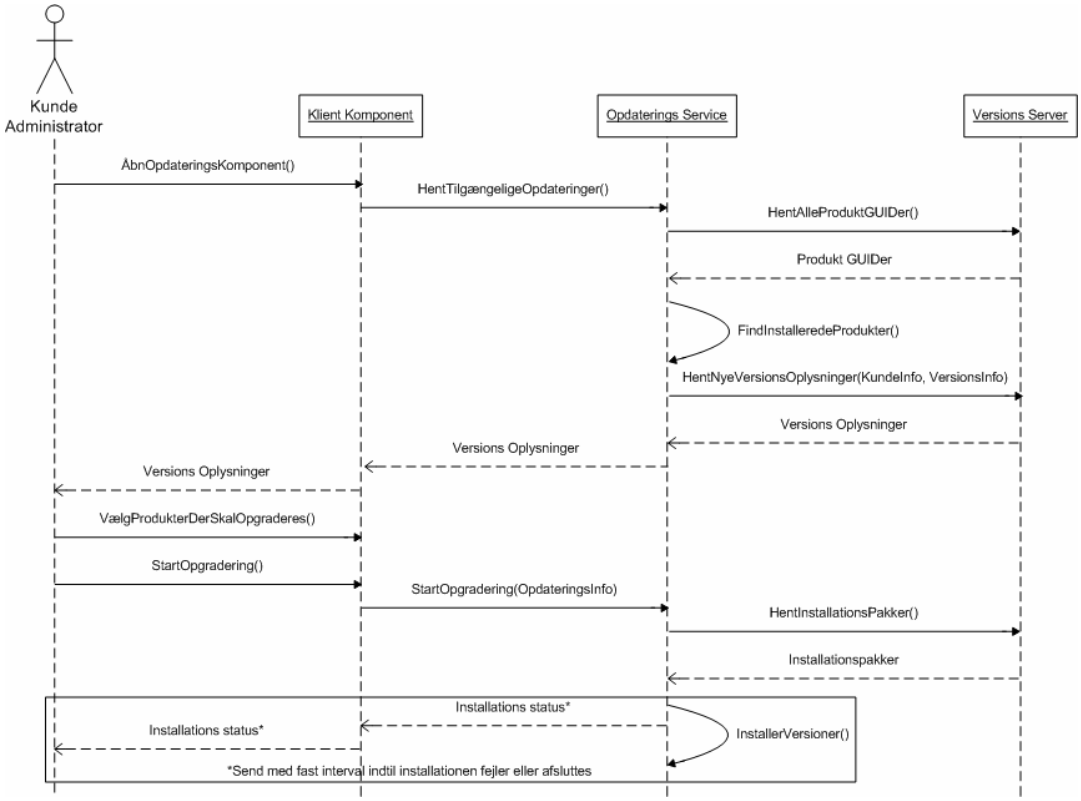
Primær aktør: Kunde Administrator.

USE CASE 1	Halvautomatisk Opgradering
Forudsætninger	• Der kan skabes forbindelse mellem Opdatering Komponenten og Opdaterings Servicen • Der kan skabes forbindelse mellem Opdaterings Servicen og Versions Serveren. • Versionsoplysninger er tilgængelige hos Kunden. • Versionerne er tilgængelige på Versions Serveren.
Succes kriterier	• Produkterne er opdaterede til de ønskede Versioner. • Opgraderingen er logged på Versions Serveren. • Kundens nye Versioner er registreret på Versions Serveren.
Hoved forløb	
1	Kunden logger sig på sin egen portal med administrator rettigheder.
2	Opdaterings Komponenten anmoder Opdaterings Servicen om tilgængelige opdateringer.
3	Opdaterings Servicen henter alle Produkt ID'er fra Versions Servicen.
4	Opdatering Servicen leder efter installerede Produkter i registrerings databasen, ved brug af de Produkt ID'er den har hentet. Kundeoplysninger, Produkt ID'er og versionsnumre returneres til Versions Servicen.
5	Versions Servicen logger kundens versionsoplysninger.
6	Versions Servicen svarer tilbage med Versionsoplysninger for de Produkter der kan opdateres.
7	Opdaterings Servicen svare tilbage til Opdaterings Komponenten, med

	Versionsoplysningerne.
8	Kunden vælger de Produkter han ønsker at opdatere.
9	Listen over valgte opdateringer sendes til Opdaterings Servicen.
10	Opdaterings Servicen henter og installerer installationspakkerne fra Versions Serveren. Kunde Administratoren præsenteres for et statusvindue, hvor han kan følge installationsprocessen.
11	Kunden præsenteres for installations resultatet.
12	Der fortsættes fra step 2.
Udvidelser	
1a	Kunde Administratoren har ikke administratorrettigheder. 1. Opdatering Komponenten vises ikke. Håndteres af WebtopONE. Use casen afsluttes.
2a	De påkrævede Kundeoplysninger er ikke registreret: 1. Kunden præsenteres for en formular hvor informationerne skal indtastes før der kan fortsættes.
8a	Der er ingen tilgængelige opdateringer. 1. Kunden informeres herom. Use casen afsluttes.
8b	Hvis kunden ønsker at ændre sine oplysninger, kan han via et link blive præsenteret for en formular der gør dette muligt. (Use Case Ia)
8c	Kunden ønsker ingen af de, tilgængelige opdateringer, 1. Use casen afsluttes.
*	Kommunikationen til Versions Serveren fejler. 1. Kommunikationen forsøges genetableret. Hvis dette lykkedes fortsættes som normalt ellers afsluttes use casen.

Da der ikke er nogle ændringer til use case Ia er denne ikke gentaget i dette afsnit. Der henvises til Analyse afsnittet for flere oplysninger.

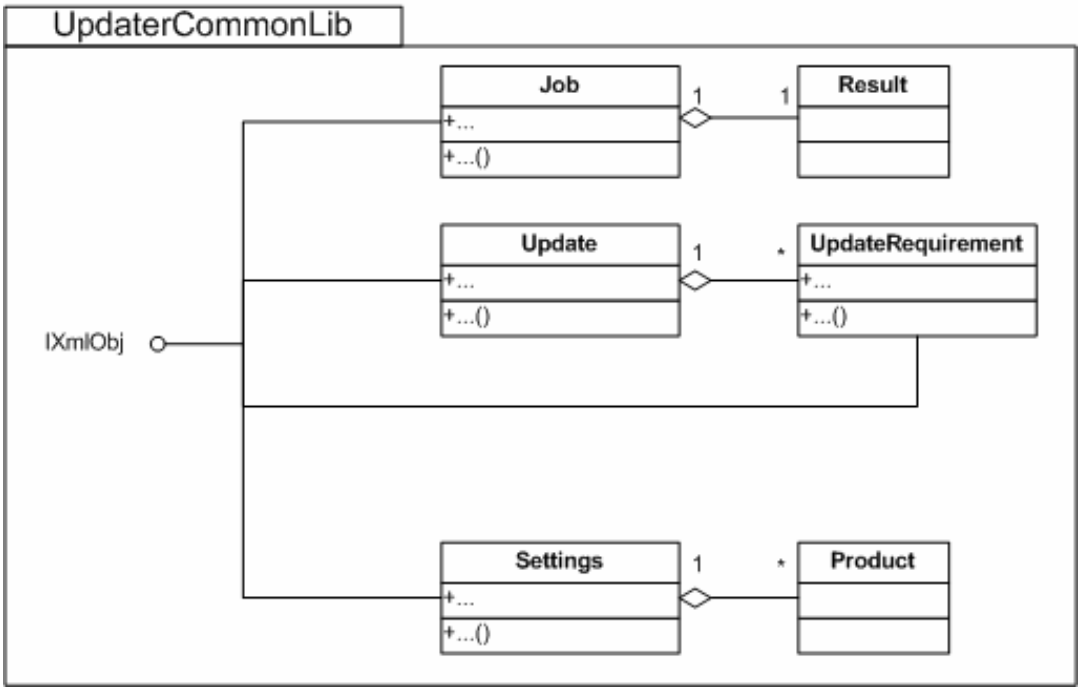
Herunder ses et sekvens diagram for et eksempel på brug af systemet af en Kunden, baseret på use case 1.



Figur 12: Sekvensdiagram for Kunde Administratorbrug

7.4.2. UpdaterCommonLib

UpdaterCommonLib bliver ligesom CommonLib brugt fra flere steder. Den er dog lidt mere specialiseret og beskæftiger sig kun med ting vedrørende selve opdatering processen.
Den indeholder strukturer, der bruges til at repræsenter et opgraderingsjob, samt resultatet af dette.
Den indeholder også en struktur, der kan repræsenter specielle oplysninger om et Produkt, heriblandt om opgradering af Produkt med systemet, skal deaktiveres.
Ligesom alle andre objekter der skal kunne sendes mellem delkomponenter, kan objekterne i UpdaterCommonLib konverteres til og fra XML.



Figur 13: UpdaterCommonLib klassebiblioteket

7.4.3. Opdaterings Service

Opdaterings Servicen har til ansvar at sørge for kommunikationen med Versions Serveren fra Kundens Serveren. Når Opdatering Komponenten har brug for Produktinformationer ”henvender” den sig til Opdaterings Servicen, som henter informationer fra Versions Serveren.
Kunden har også mulighed for ved hjælp af AWOConfiguratoren, at foruddefinerer hvornår de fuldautomatiske opdateringer, som håndteres af denne service, skal køres.

De enkelte delelementer i servicen er beskrevet herunder.

InstallFSM
Installations processen inderholder forskellige trin, der kræver brugen af eksterne programmer. For at kunne holde styr på hvor langt processen er nået og hvad der skal ske bagefter, har vi brug for en FSM (finite state machine), til at styre forløbet.

State machinen består af 5 states:

***CollectInfo*:**
Indsamler eventuelle oplysninger, som er nødvendige for at kunne installerer en given installationspakke.
Da der i skrivende stund ikke forligger nogen endelig dokumentation, for hvilke oplysninger der er nødvendige for installationspakkerne i den nye Version 5.0 af WebtopONE, har denne state endnu ingen funktion, men er medtaget for senere let at kunne blive implementeret, når disse oplysninger bliver defineret.

Download:
Overfører ved hjælp af BITSdownloaderen de installationspakker, som skal installeres fra Versions Serveren.
Når alle installationspakker er blevet overført skiftes til næste state ”Install”.

Install:
Starter de forskellige installations pakker som skal installeres, en efter en, som en ny process.
Det er i skrivende stund på grund af begrænsninger i InstallAware, ikke muligt at følge med i processen af selve installationen, når disse køres silent. Derfor vil

attributten ”Progress”, som specificerer hvor langt installationen er i en given state, ikke være sat for denne state.
Når alle installationspakker er blevet kørt, skiftes til næste state ”Finish”.

Finish
Rydder op efter installationen. Dvs. sletter de midlertidige installationsfiler som er blevet hentet under opdateringsprocessen.
Derudover sættes de forskellige attributter, som angiver hvilke Installationer som gik godt og hvilke der fejlede.
Da dette er den sidste state, vil statemachinen forblive i denne state, indtil en ny opdateringsprocess sættes i gang.

Error
Under overførelsen af Installationspakkerne fra Versions Serveren til Kunde Serveren, er der en lang række ting som kan gå galt og stoppe overførelsen. Hvis disse fejl er uoprettelige, vil der blivet skiftet til denne state, så Brugeren kan blive informeret om problemet og rette dette, inden opgraderingen forsøges igen.

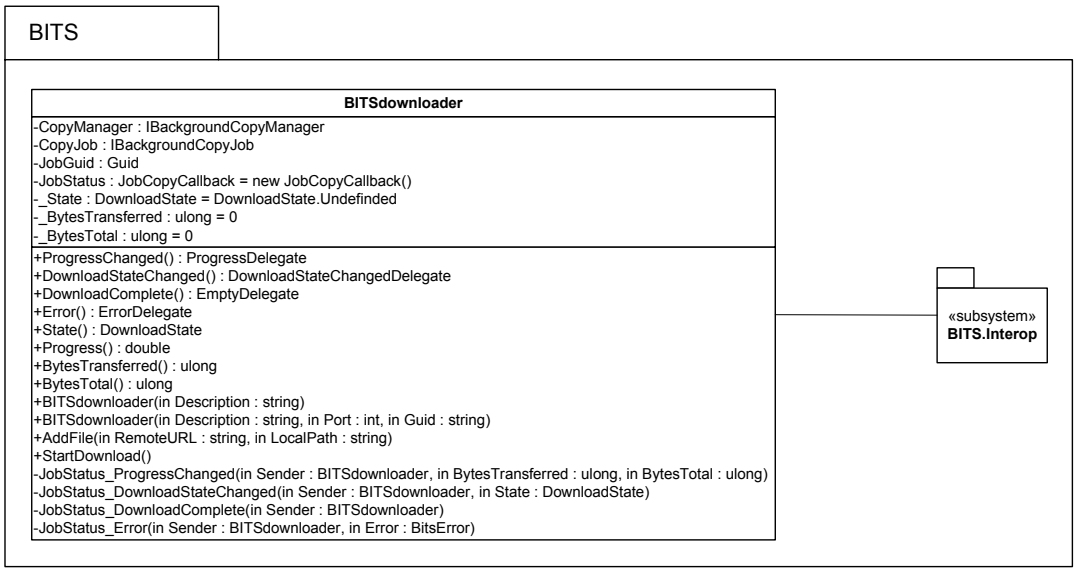
Figur 14: InstallFSM

BITS
Til overførelsen af installationspakkerne fra Versions Serveren til Kunde Serveren, anvendes Windows egen indbygget WindowsService: BITS.

BITS har en lang række indbygget features, som AWO kan gøre brug af:

- Kan pause aktive downloads i tilfælde af at forbindelsen mistes, computeren genstartes og lignende.
- Kan sættes til kun at bruge ubenyttet båndbredde, så downloads ikke generer anden netværkstrafik.
- Veldefinerede fejlbeskeder, hvis noget går galt under en overførelse.

BITS bruges normalt af Windows Update og lignede services som skal overføre større mængder data pålideligt over HTTP og er dermed oplagt at bruge for AWO.
På nuværende tidspunkt er BITS unmanaged kode. Dvs. al kommunikation med servicen forgår via besværlige ComInterfaces, som tager kryptiske strukturer og enumurations som parameter i form af pointerer.
For at gøre brugen af BITS lettere tilgængelig for os selv og eventuelt andre i webtop, som senere skal anvende servicesen, har vi indkapslede den forholdsvise besværlige kommunikation, i en lettilgængelig .NET klasse, indeholdende en række funktioner, attributter, events og nestede klasser, som implementerer de funktionaliteter, som vi ønsker at anvende fra servicen.



Figur 15: BITS klassebiblioteket

Webserver
Under selve opgraderingen af Produkterne, kan det være nødvendigt at stoppe eller genstarte IIS'en. For at kunne bevare forbindelsen med Opdaterings Servicen skal der bruges en webserver, der kan overtage kommunikationen. Den skal understøtte HTTP GET som minimum.
Vi har selv implementeret en sådan simpel webserver, der kan vise hvor langt installationsprocessen er nået, samt præsentere en oversigt over hvilke opgraderinger der lykkedes og hvilke der fejlede.
Selve GUIen som bliver præsenteret for brugeren, er en HTML side, som ved hjælp af javscript og ajax henter informationer om installationsprocessen fra Install-FSMen gennem Webserveren

7.4.4. Opdaterings Komponent (Webtop komponent)

Opdatering Komponenten er Kunde Administratorens interface til at foretage halv-automatiske opdatering af hans WebtopONE produkter.
Halvautomatisk opdatering betyder, at Kunde Administratoren selv vælger hvilke Produkter der skal opgraderes til den nyest tilgængelige version. Systemet finder selv ud af hvilke Produkter der er installeret og hvilken Version der kan opgraderes til, se afsnit 7.2.3.

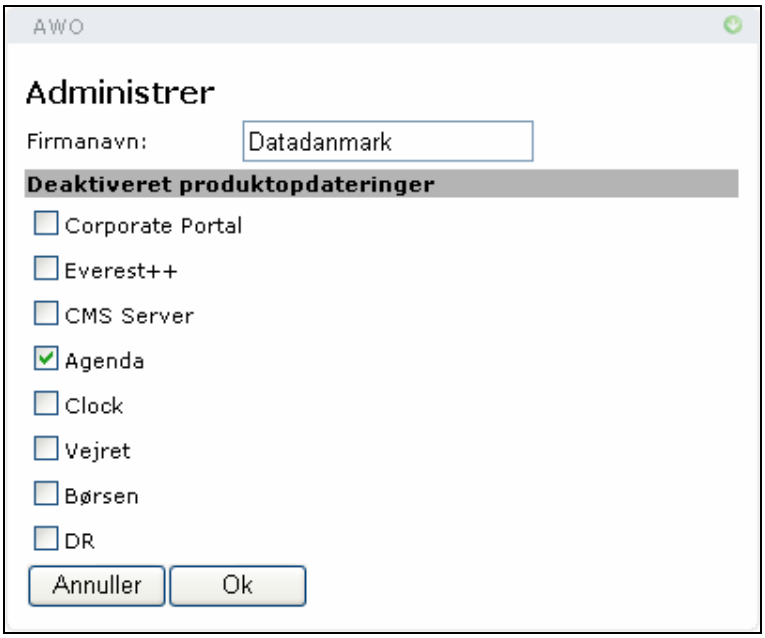
Opdatering Komponenten er henvendt til den eller de personer hos Kunden der er ansvarlig for vedligeholdelse af firmaets software. Da det vil være de samme personer der skal benytte komponenten hver gang, er der ikke en længere beskrivelse af hvordan, systemet skal bruges på selve komponenten. Vi har også begrænset de informationer, der umiddelbart kan ses til det nødvendige, med mulighed for et se flere ved få klik. Da komponenten skal placeres på en WebtopONE portal, sammen med mange andre komponenter, er det også en fordel at den optager så lidt plads som muligt.
Vi har designet GUI'en ud fra principper baseret på interaktions design og derfor bl.a. fjernet unødige design-elementer, så som rammer o.l.

Opdaterings Komponenten er realiseret som en Webtop komponent, der kan sættes ind på en Kundes portal.

Figur 16: Halvautomatisk opdatering med Opdaterings Komponenten

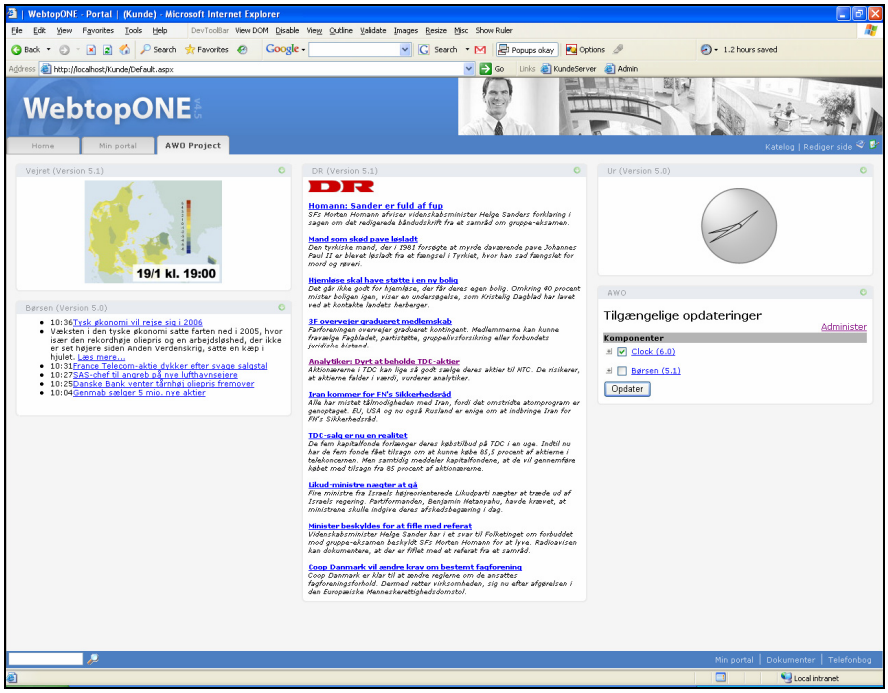
Hver Kategori vises som en overskrift med fed på mørkegrå baggrund.
De Produkter der kan opgraderes, vises så herunder med en checkbox, der afgør om de skal opdateres.
Det lille plus-ikon åbner, ligesom linket på navnet og en beskrivelse med flere oplysninger om versionen.

Det er også muligt at styre, hvilke Produkter det skal være muligt at opdatere, samt firmaets navn ved at klikke på Administrer linket, der åbner nedenstående vindue.

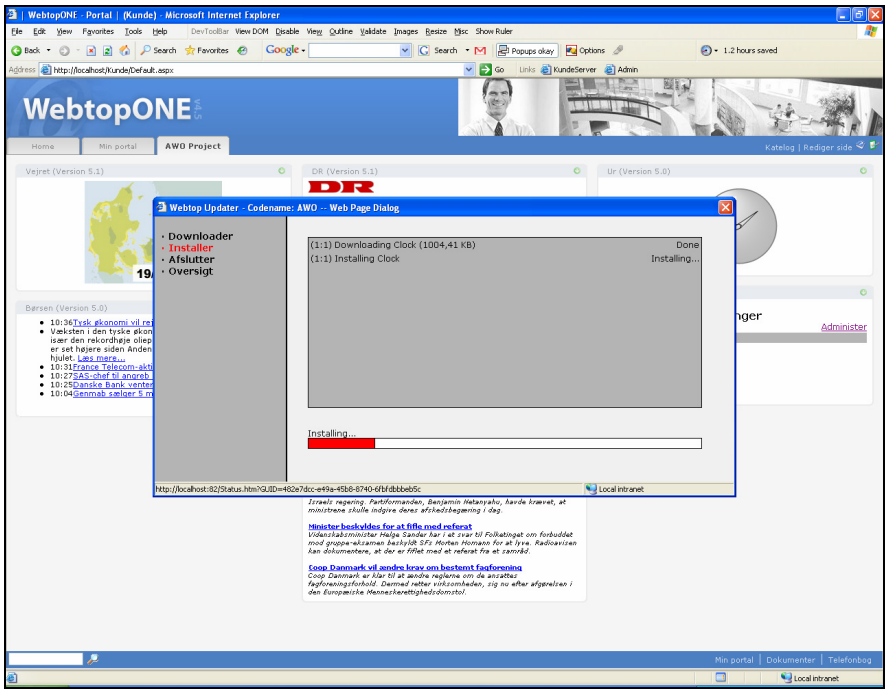


Figur 17: Deaktivering af produktopdateringer i Opdaterings Komponenten

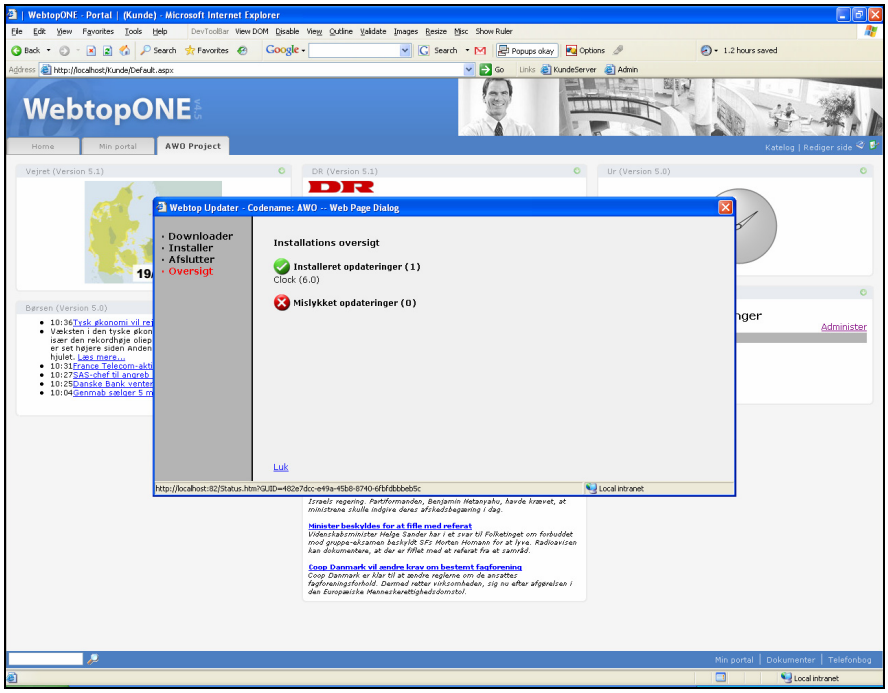
På de følgende sider er der vist et eksempel på Opdaterings Komponenten i brug.



Figur 18: Opdaterings Komponenten sat ind på en portal



Figur 19: Opdaterings processen er påbegyndt. Processen kan følges i status vinduet

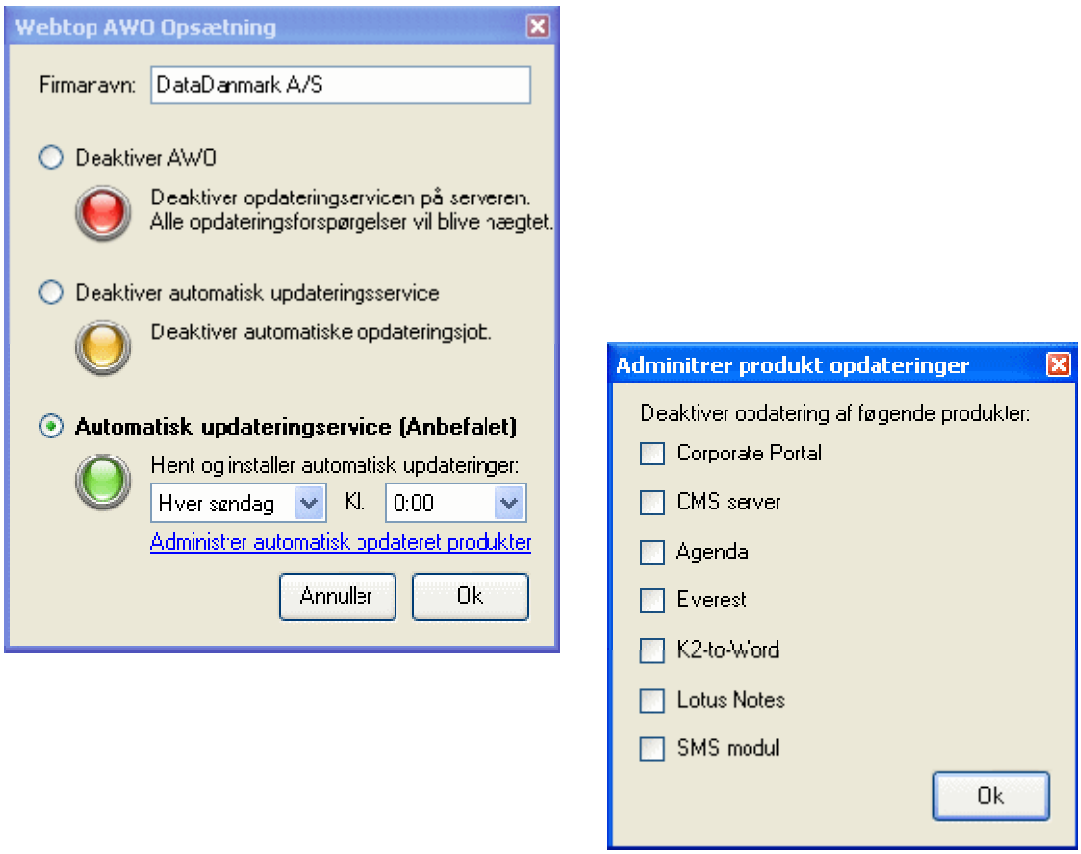


Figur 20: Installations processen er nu afsluttet. Resultatet præsenteres i status vinduet

7.4.5. AWOConfigurator

AWOConfigurator er en Windows applikation, hvis formål er at gøre det muligt for Kunde Administratoren at ændre instillingerne for Opdaterings Servicen.
Der er 3 typer instillinger:

- Kundens informationer, pt. firmanavn.
- Opdaterings Servicen kan sættes til deaktiveret, halvautomatisk opdatering, eller fuldautomatisk opdatering.
- Opgraderingen af installerede Produkter kan deaktiveres.



Figur 21: Screenshot af AWOConfiguratoren

7.5. Webtop Server

Webtop Serveren er den WebtopONE server, hvor Webtop har sin portal. Det er her Administrations Komponenten kan indsættes på en portal fra det digitale katalog over WebtopONE komponenter.

7.5.1. Use case 2 – Administrér Versions Server

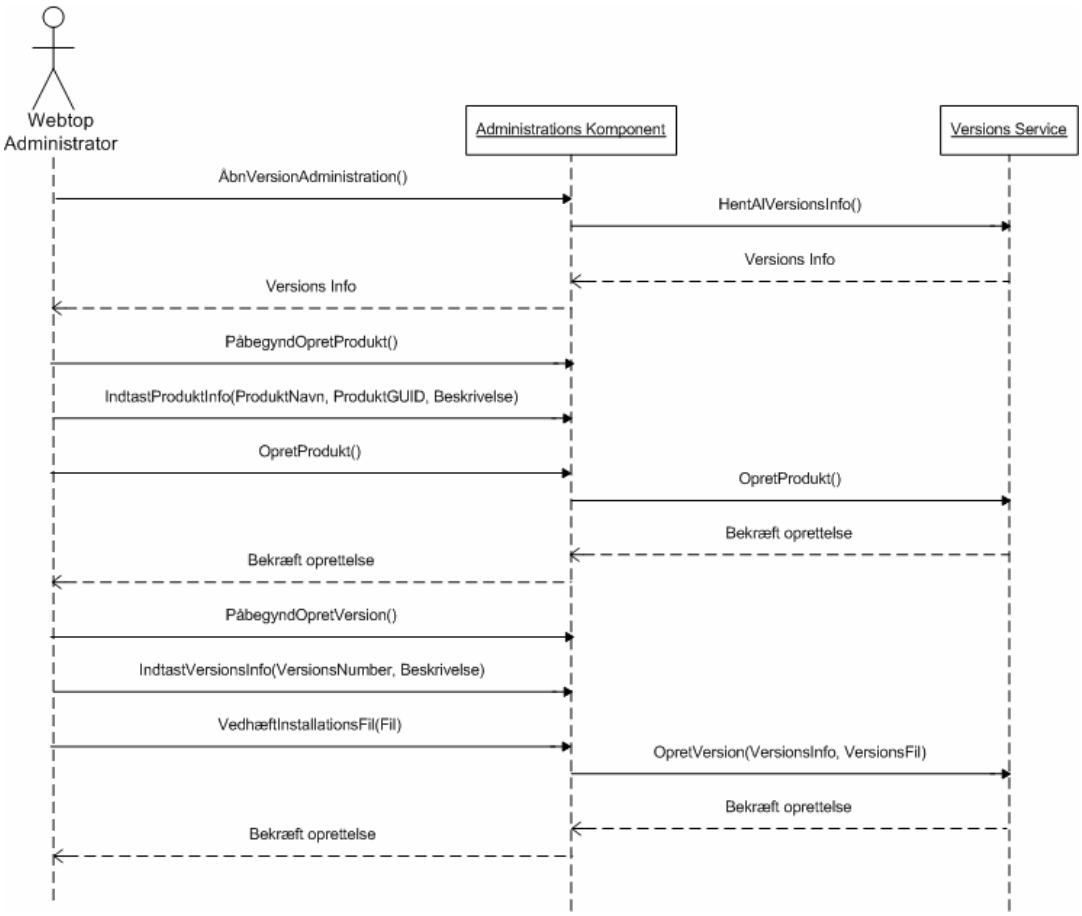
Nedenstående use case er en teknisk uddybelse af overordnet usecase II fra analyse afsnittet. Den beskriver hvordan en Webtop Administrator kan vedligeholder Versions Serveren via Administrations Komponenten

Primær aktør: Webtop Administrator.

USE CASE 2	Administrér Versions Server
Forudsætninger	• Administratoren er identificeret og bekræftet. • Der kan skabes forbindelse mellem Administrations Komponenten og Versions Servicen.
Succes kriterier	• De ønskede ændringer er blevet registreret.
Hoved forløb	
1	Administratoren logger på sin portal.
2	Administratoren foretager de ønskede ændringer.
Udvidelser	
2a	Kontrol af Kunder oplysninger (Use Case IIa)
2b	Tilføj Kategori (Use Case IIb)
2c	Rediger Kategori (Use Case IIc)
2d	Slet Kategori (Use Case IId)
2e	Tilføj Produkt (Use Case IIe)
2f	Rediger Produkt (Use Case IIf)
2g	Slet Produkt (Use Case IIg)
2h	Tilføj Version (Use Case IIh)
2i	Rediger Version (Use Case IIi)
2j	Slet Version (Use Case IIj)
*	Kommunikationen til Versions Serveren fejler. 1. Kommunikationen forsøges genetableret. Hvis dette lykkedes fortsættes som normalt ellers afsluttes use casen.

Da der ikke er nogle ændringer til use case IIb-IIj, er disse ikke gentaget i dette afsnit. Der henvises til Analyse afsnittet for flere oplysninger.

Herunder ses et sekvens diagram for et eksempel på brug af systemet af Kunden, baseret på use case 2, IIe og IIh.



Figur 22: Sekvensdiagram for Webtop Administratorbrug

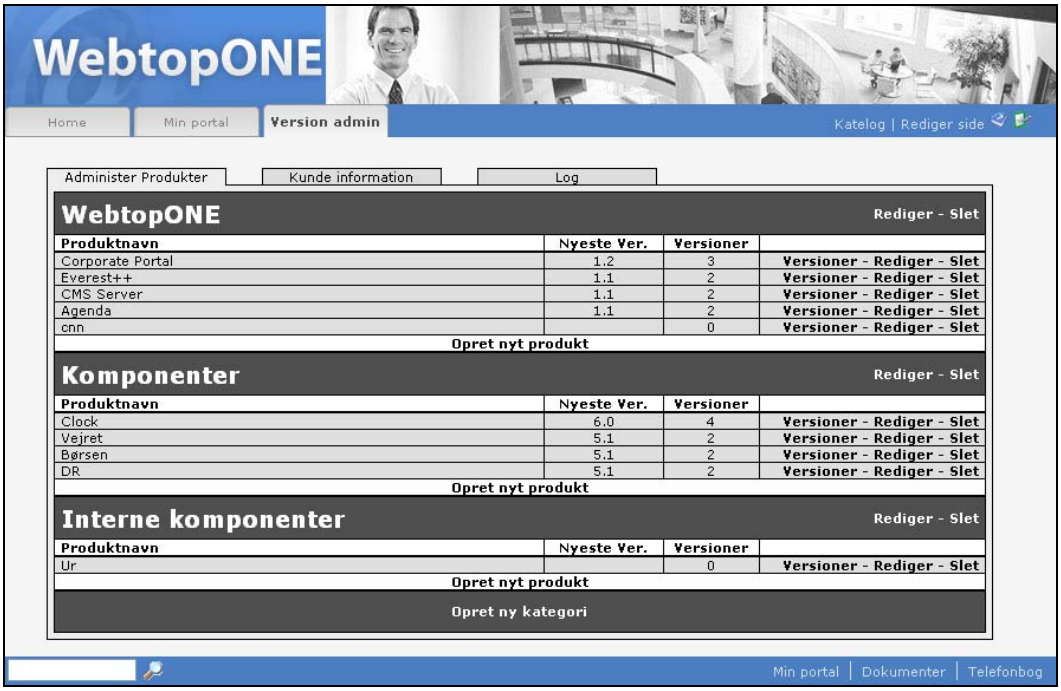
7.6. Administrations Komponenten (Webtop komponent)

Administrations Komponenten gør det muligt for de ansatte i Webtop, at vedligeholde Versions Serveren, således at Kunderne altid har de nyeste opgraderinger tilgængelige.

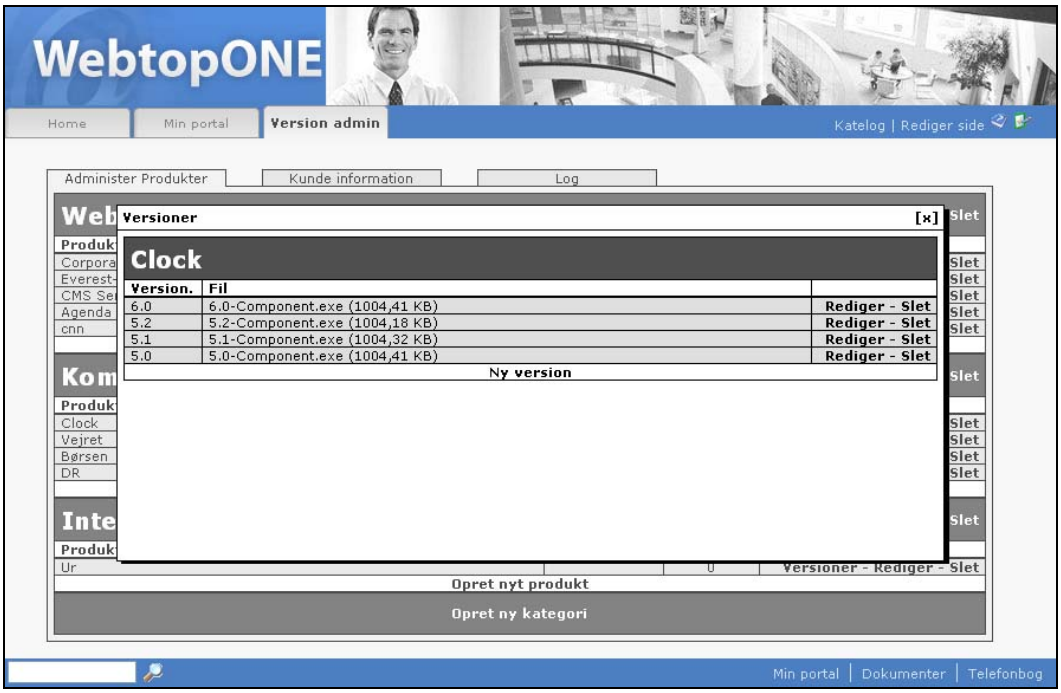
Komponenten indeholder 3 hovedvinduer der kan vise ved brug af faneblade; Administrer Produkter, Kunde Information og Log.
Det første, Administrer Produkter, er det primære vindue, der som navnet hentyder, gør det muligt at administrere Katekorier, Produkter og Versioner.
Ligesom på Opdaterings Komponenten står kategorierne på mørkegrå baggrund. Under hver Kategori vises de produkter der er tilknyttet denne, samt et link der gør det muligt at tilføje nye produkter.
De funktioner der er tilknyttet et bestemt punkt (Kategori, Produkt, Version) er vist i den samme vandrette række som punktet.
Alle ændringer foretaget af Administratoren, bliver via Versions Servicen, gemt i filen, Products.xml. Da det også er Versions Servicen der informerer Opdaterings Servicen om hvilke Versioner der er tilgængelige, afspejles dette øjeblikkeligt.

Der er benyttet ”lag” i form a tags, med absolut position, til at lave ”vinduer” til indtastning o.l. Derudover ”dimmes” baggrunden, så det bliver nemmere at se, hvad der er det aktive indhold på siden.

På den følgende side er vist 2 screenshots af Administrations Komponenten.



Figur 23: Administrations Komponenten



Figur 24: Administrations Komponenten, med versionsvinduet åbent

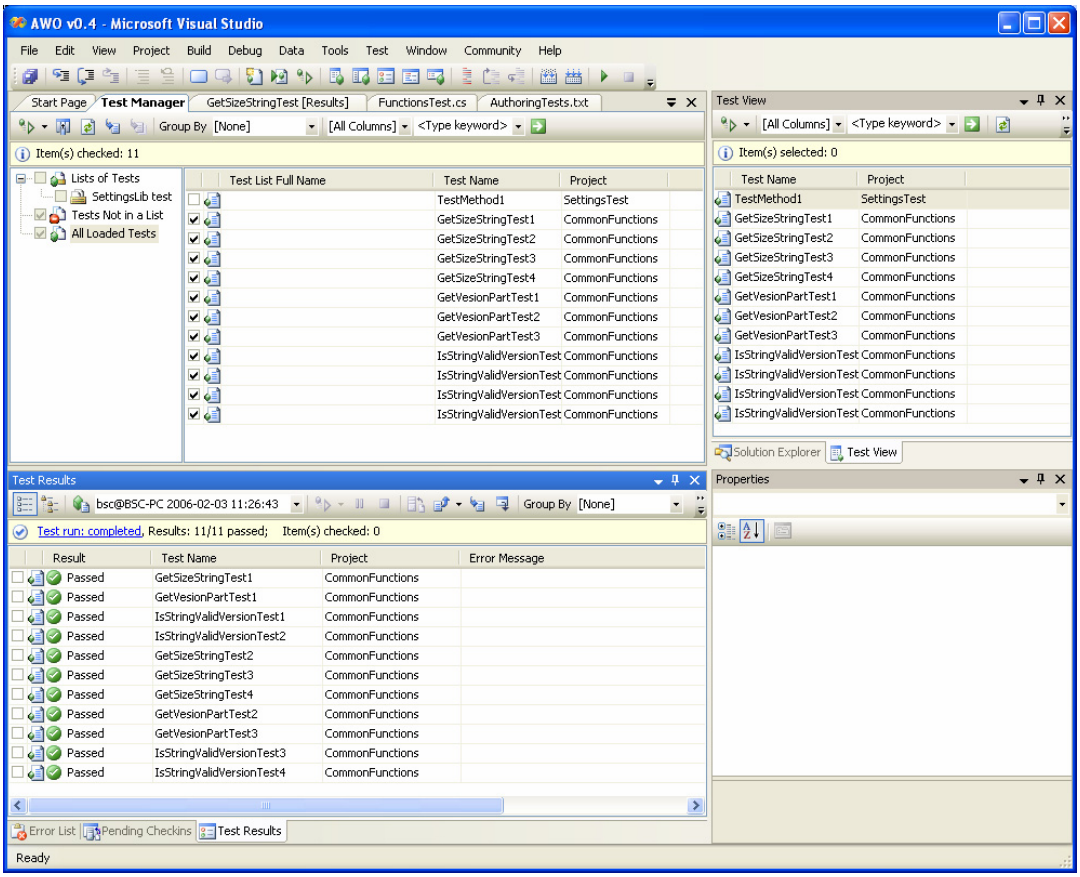
KAPITEL 8

8. Test (Evaluering)

For at undgå at skulle vente på at store ”rigtige” installationspakker blev installeret under udvinklingen, har vi konstrueret en række ”micro”-installationspakker, der basalt set kun ”installerer” en tekstfil med oplysninger om Produktet. Derudover registrerer den oplysninger i registreringsdatabasen, på samme måde som de Produkter der skal bruges i den endelige løsning.
Vi har også kunne teste, ved at kigge i de generede XML filer, da strukturen gør oplysningerne umiddelbart læselige.
Da hvert element er ansvarlig for sin del af koden og kommunikations formen var fastlagt på et tidligt tidspunkt, har der ikke været de store problemer med at sammensætte elementerne.
Det delelement der umiddelbart var mest besværligt at teste, var Opdaterings Servicen. Da dette er en Windows Service, er den eneste umiddelbare mulighed, at skrive til event loggen. For at komme over dette problem, har vi placeret logikken til Servicen i en separat dll. Når servicen startes, benytter den dll’en til at starte en række tråde. Denne struktur gør at vi under udviklingen, kan erstatte servicen med en konsolapplikation, der så kan skrive debug information direkte til konsollen.

8.1. Unit Test

Under udviklingen af delelementerne, har den ansvarlige udvikler for de enkelte elementer, selv sørget for løbende at lave unit test.
Til unit testene har vi så vidt det har været muligt, anvendt den indbygget Unittest i Visual Studio 2005 Team Suiten.
Unit testene bliver lavet som en række klasser, med funktioner som kalder den eller de dele som skal testes, med en række foruddefineret argumenter og forventede retur værdier.
Når Unittestene køres, kaldes disse funktioner og de forventede returværdier bliver sammenlignede med de faktiske returværdier.



Figur 25: Screenshot af unittest i Visual Studio Team Suite

Det er desvære ikke alle delelementer, som det har været muligt at teste på denne måde, da nogen funktioner er afhængige af store avanceret objekt strukturer, asynkrone kald, input fra andre externe komponenter osv. For at teste disse funktioner, har vi istedet anvendt debuggeren i Visual Studio, til at stoppe eksekveringen af kode og selv manuelt undersøgt om funktionerne opførte sig som forventede

8.2. Use case test

Efterhånden som de forskellige delelementer blev færdige, har vi udført decideret use case tests, for at teste om AWO systemet lever op til de opsatte krav.
Til hver use case test er blevet udarbejdet en Testplan, som beskriver hvordan testen skal udføres, hvilke forudsætninger der skal være opfyldt, for at testen kan udføres, tilladte afvigelser/fejl, samt nødvendige handlinger ved fundne fejl.
Til hver testplan hører også en testplan, som dokumenterer resultatet af den udførte test.

De 4 vigtigste use case tests er inkluderet i rapporten på de følgende sider, de resterende kan findes under bilag 12.1

Tests er inkluderet i rapporten:

Testplan I – Opgradering af Produkter
Testplan Ia – Ændring af kundeoplysninger
Testplan IIa – Kontrol af kundeoplysninger
Testplan IIh – Tilføj ny Version

TESTPLAN

Use case I – Opgradering af Produkter

Testplan ID.:	**version.:**	**Testtype:**	**Dato:**	**Tester:**
1	1.1	Use case test	16/1 2006	Bo S. Christensen

Testformål:
Teste funktionaliteten beskrevet i use case I – Opgradering af Produkter.

Anden information:
Da der endnu ikke findes nogen færdige komponenter til version 5.0 af WebtopONE, er der blevet udviklede 4 komponenter, i 2-3 versioner med dertilhørende installere, specielt til AWO projektet.
Disse komponenter har alle deres versionsnummer stående til højre for deres navn, så det er let at se om en opdatering er blevet gennemført korrekt.

Testforudsætninger:
- En tidligere version af Produktet skal allerede været installeret
- En nyere version af et installeret Produkt skal findes på Versions Serveren (opret evt. en ny version, se use case IIh)
- Versions Serveren skal køre og der skal kunne skabes forbindelse til denne.
- Opdaterings Servicen skal køre.

Udførsel:
1. Brugeren logger på sin WebtopONE portal (Kunde Serveren)
2. Brugeren vælger et eller flere Produkter, som skal opgraderes og trykker opdater.
3. Brugeren præsenteres for et nyt vindue, hvori installationsprocessen kan følges.
4. Når installationen er færdig, vises en installationsoversigt.
5. Installationsoversigten lukkes og brugeren præsenteres for en opdateret liste over tilgængelig opdateringer.
6. Brugeren validere manuelt om opgraderingen er blevet fuldført, ved at tjekke versions nummeret på komponenten.

Korrigerende handlinger ved fundne fejl:
Fejl rapporteres til den ansvarlige udvikler for delkomponenten, så denne kan blive rettet. Hvorefter testen udføres igen.

TESTRAPPORT

Use case I – Opgradering af Produkter

Testrapport ID:	**version:**	**Testtype:**	**Udført:**	**Tester:**
1	1.1	Use case test	16/1 2006	Bo S. Christensen

Testresultat:

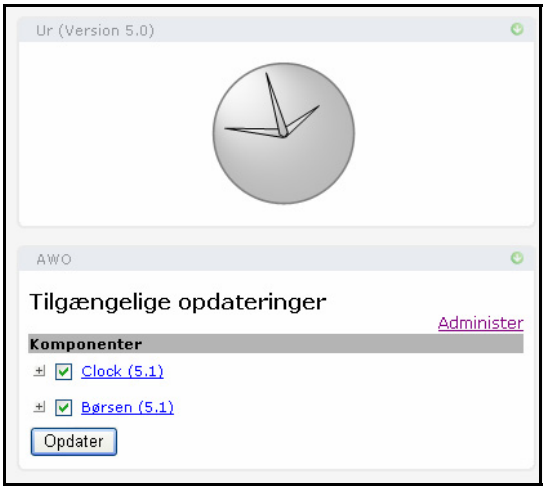


Fig 1 - Før opdatering

Fig 2 - Efter opdatering

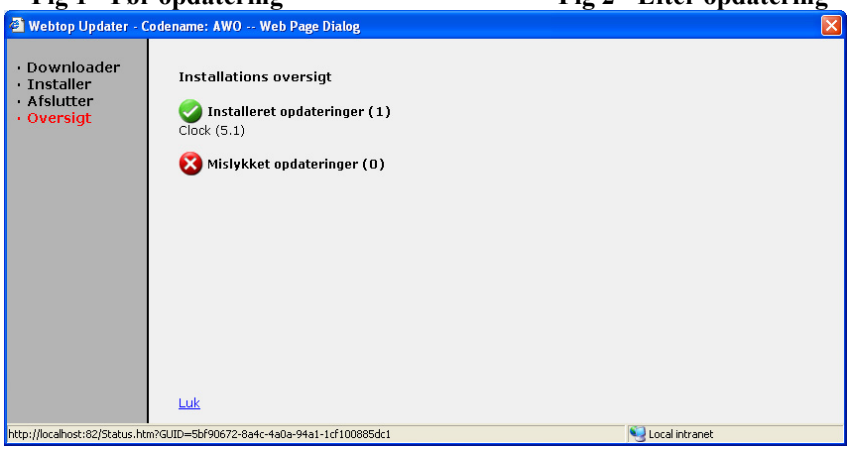


Fig 3 - Installationsoversigt

Konklusion:

På ovenstående figurer ses ”ur” komponenter før og efter opdateringen.
Som det kan ses på versionsnummerne til højre for komponentoverskriften og udfra installationsoversigten, er opdateringen gennemført korrekt

TESTPLAN

Use case Ia – Ændring af kundeoplysninger

Testplan ID.:	**version.:**	**Testtype:**	**Dato:**	**Tester:**
1a	1.0	Use case test	17/1 2006	Bo S. Christensen

Testformål:
Teste funktionaliteten beskrevet i use case Ia – Ændring af kundeoplysninger.

Anden information:
Kundeoplysninger gemmes lokalt på Kundens Server i filen AWOSettings.xml og på Versions Serveren i filen Customers.xml

Testforudsætninger:
- Versions Serveren skal køre og der skal kunne skabes forbindelse til denne.
- Opdaterings Servicen skal køre.

Udførsel:
1. Brugeren logger på sin WebtopONE portal (Kunde Serveren).
2. Brugeren klikker på ”administrer” knappen.
3. Firmanavn udfylders og Produkter aktiveres/deaktivers.
4. Brugeren klikker på ”ok” knappen.
5. Brugeren validerer manuelt at oplysningerne er blevet gemt korrekt i de 2 filer: Customers.xml og AWOSettings.xml, på henholdsvis Versions Serveren og Kunde Serveren.

Korrigerende handlinger ved fundne fejl:
Fejl rapporteres til den ansvarlige udvikler for delkomponenten, så denne kan blive rettet. Hvorefter testes udføres igen.

TESTRAPPORT

Use case Ia – Ændring af kundeoplysninger

Testrapport ID:	**version:**	**Testtype:**	**Udført:**	**Tester:**
1a	1.0	Use case test	17/1 2006	Bo S. Christensen

Testresultat:

Indtastede oplysninger:

Firmanavn: DataDanmark A/S

Deaktiveret produkter: Agenda

AWOSettings.xml:

```
<Settings>
 <CompanyName>Datadanmark</CompanyName>
 <CustomerGuid>ade7b144-f67f-456c-acd4-da933c6494c3</CustomerGuid>
 <ServiceState>2</ServiceState>
 <UpdateDay>0</UpdateDay>
 <UpdateHour>13</UpdateHour>
 <DisabledProducts>
  <Product>
   <ProductName>Agenda</ProductName>
   <ProductGUID>f09f4d52-77f9-482b-9786-1538d2a1e48f</ProductGUID>
  </Product>
 </DisabledProducts>
</Settings>
```

Customers.xml:

```
<Customers>
 <Customer Name="Datadanmark" GUID="ade7b144-f67f-456c-acd4-
da933c6494c3">
  <InstalledProducts LastUpdated="16-01-2006 13:48:00">
   <VersionReference>
    <VersionNumber>1.1</VersionNumber>
    <ProductGUID>f09f4d52-77f9-482b-9786-1538d2a1e48f</ProductGUID>
   </VersionReference>
...
  </InstalledProducts>
 </Customer>
</Customers>
```

Konklusion:

Som man kan se på indholdet af de 2 filer, er Firmanavn blevet gemt korrekt på både kundens server og på versionsserveren. Oplysninger om det deaktiveret produkt (Agenda) er også blevet gemt korrekt på kundeserveren.

TESTPLAN

Use case IIa – Kontrol af kundeoplysninger

Testplan ID.:	**version.:**	**Testtype:**	**Dato:**	**Tester:**
2a	1.0	Use case test	17/1 2006	Bo S. Christensen

Testformål:
Teste funktionaliteten beskrevet i use case IIa – Kontrol af kundeoplysninger.

Anden information:
Hver gang en kunde opdaterer et produkt, logges det på versionsserveren i filen customers.xml og kan ses via administrationskomponenten.

Testforudsætninger:

- Versionsserveren skal køre og der skal kunne skabes forbindelse til denne.

Udførsel:

1. Log på Kundens WebtopONE portal.
2. Noter Firmanavn og hvilke produktversioner som er installeret.
3. Log på Webtops WebtopONE portal (Webtop Server).
4. Vælg fanebladet ”kunde information”.
5. Vælg kunden som blev brugt i step 1.
6. Valider manuelt om data fra step 5 og 2 stemmer overens.

Korrigerende handlinger ved fundne fejl:
Fejl rapporteres til den ansvarlige udvikler for delkomponenten, så denne kan blive rettet. Hvorefter testen udføres igen.

TESTRAPPORT

Use case IIa – Kontrol af kundeoplysninger

Testrapport ID:	**version:**	**Testtype:**	**Udført:**	**Tester:**
2a	1.0	Use case test	17/1 2006	Bo S. Christensen

Testresultat:

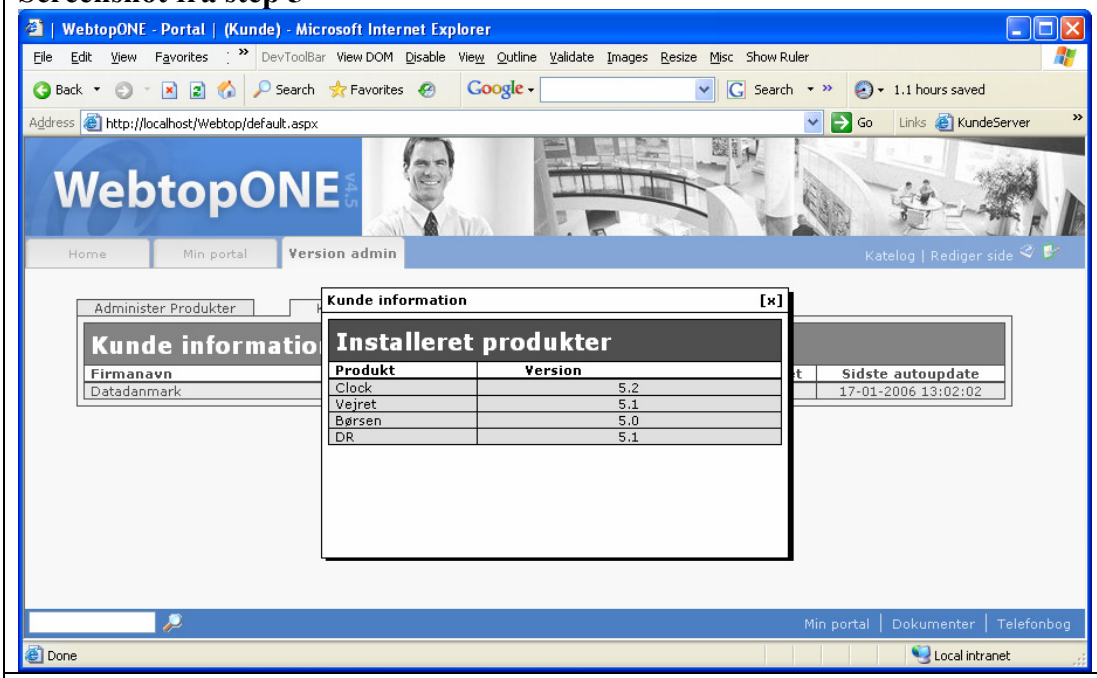
Noteret information:

Firmanavn: DataDanmark

Installede produkter:

- Clock 5.2
- Vejret 5.1
- Børsen 5.0
- DR 5.1

Screenshot fra step 5



Konklusion:

Som man kan se på screenshottet fra step 5, stemmer information om firmanavn og installeret produkter, overens med data noteret i step 2.

TESTPLAN

Use case IIb – Tilføj ny Kategori

Testplan ID.:	**version.:**	**Testtype:**	**Dato:**	**Tester:**
2b	1.0	Use case test	18/1 2006	Bo S. Christensen

Testformål:
Teste funktionaliteten beskrevet i use case IIb – Tilføj ny Kategori

Anden information:
Information om Versioner, Produkter og Kategorier bliver gemt i filen Products.xml på VersionsServeren.

Testforudsætninger:
- Versions Serveren skal køre og der skal kunne skabes forbindelse til denne.

Udførsel:
1. Log på Webtops WebtopONE portal (Webtop Server).
2. Tryk på ”opret ny kategori”.
3. Indtast et navn for den nye Kategori og tryk ok.
4. valider manuelt at Kategorien er blevet oprettet korrekt i Products.xml

Korrigerende handlinger ved fundne fejl:
Fejl rapporteres til den ansvarlige udvikler for delkomponenten, så denne kan blive rettet. Hvorefter testen udføres igen.

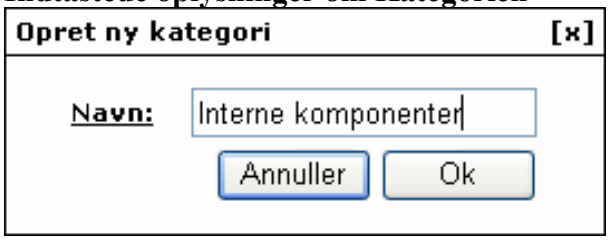
TESTRAPPORT

Use case IIb – Tilføj ny Kategori

Testrapport ID:	**version:**	**Testtype:**	**Udført:**	**Tester:**
2b	1.0	Use case test	18/1 2006	Bo S. Christensen

Testresultat:

Indtastede oplysninger om Kategorien



Udsnit fra Products.xml

```
<ProductCategories>
 ...
 <ProductCategory>
  <Name>Interne komponenter</Name>
  <GUID>bcab90ca-72d0-447b-b7ae-dedb101474ca</GUID>
  <Products />
 </ProductCategory>
</ProductCategories>
```

Konklusion:

Som man kan se fra indholdet af Products.xml, stemmer informationen overens med det indtastede.

TESTPLAN

Use case IIh – Tilføj ny Version

Testplan ID.:	**version.:**	**Testtype:**	**Dato:**	**Tester:**
2h	1.0	Use case test	18/1 2006	Bo S. Christensen

Testformål:
Teste funktionaliteten beskrevet i use case IIh – Tilføj ny Version.

Anden information:
Information om Versioner, Produkter og Kategorier bliver gemt i filen Products.xml på VersionsServeren.

Testforudsætninger:

- Versions Serveren skal køre og der skal kunne skabes forbindelse til denne.

Udførsel:

1. Log på Webtops WebtopONE portal (Webtop Server).
2. Vælg et Produkt på listen eller opret et nyt (se use case IIe).
3. Klik på versioner udfor den valgte produkt.
4. Indtast oplysninger om den nye Version, tilføj installationsfil og eventuelle installationskrav.
5. Tryk ok og tjeck om den nye Version bliver tilføjet til versionslisten.
6. Valider at versionsinformationerne er gemt korrekt i Products.xml

Korrigerende handlinger ved fundne fejl:
Fejl rapporteres til den ansvarlige udvikler for delkomponenten, så denne kan blive rettet. Hvorefter testen udføres igen.

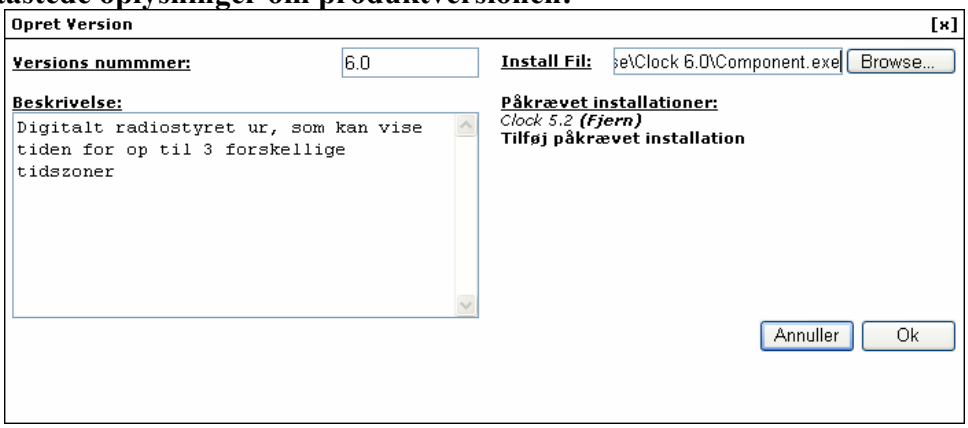
TESTRAPPORT

Use case IIh – Tilføj ny Version

Testrapport ID:	version:	Testtype:	Udført:	Tester:
2h	1.0	Use case test	18/1 2006	Bo S. Christensen

Testresultat:

Indtastede oplysninger om produktversionen:



Udsnit fra Products.xml

```
<Version>
 <VersionNumber>6.0</VersionNumber>
 <Active>True</Active>
 <GUID>31cadad9-a90e-4bd7-8d9e-28b44bf18c76</GUID>
 <Description>Digitalt radiostyret ur, som kan vise tiden for op
til 3 forskellige tidszoner</Description>
 <FilePath>http://localhost/version/data/ 6.0-
Component.exe</FilePath>
 <FileSize>1028513</FileSize>
 <CommandArgument>/s</CommandArgument>
 <RequiredVersions>
  <VersionReference>
   <VersionNumber>5.2</VersionNumber>
   <ProductGUID>D45F2999-D733-432D-B864-D882BCC03FE6</ProductGUID>
  </VersionReference>
 </RequiredVersions>
</Version>
```

Konklusion:

Som man kan se fra indholdet af Products.xml, stemmer informationen overens med det indtastede.

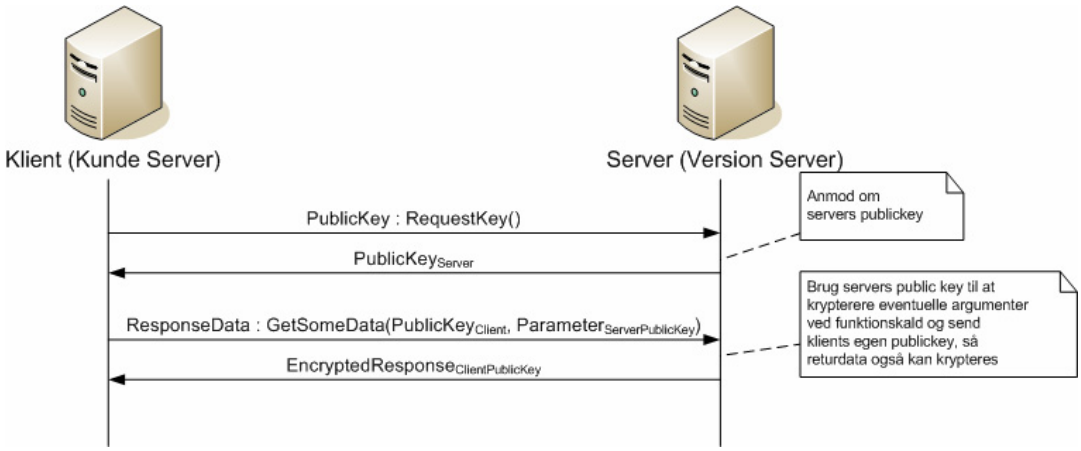
KAPITEL 9

9. Udvidelsesmuligheder

Gennem projektforløbet er vi støt på flere udvidelsesmuligheder, der kunne være brugbare i en endelig version af dette produkt. Disse emner er beskrevet i det følgende afsnit.
Herudover er de dele af projektet der ikke er implementeret, men er i produkt-specifikationen også beskrevet her.

9.1. Kryptering

I den nuværende version er den eneste oplysning om Kunden der bliver overført til Versions Serveren, firmaets navn. På længere sigt kunne det blive aktuelt at overfører andre, mere følsomme oplysninger. Hvis det er tilfældet, ville det være nødvendigt at krypterer disse oplysninger.
Man kan også argumentere for, at det er en følsom oplysning, hvilke Produkter en Kunde har installeret, i hvilket tilfælde disse oplysninger også burde krypteres.
I projektforløbet har vi konstrueret et class library, der gør det nemt at benytte asymmetrisk RSA kryptering, der vil kunne anvendes til dette formål.
Vi har dog valgt ikke at implementere selve krypteringen, da den ikke er vital for at løsningen kan testes.
Nedenstående figur beskriver, hvordan krypteringen af de følsomme oplysninger kan krypteres.



Figur 26: Diagram over RSA kryptering

På overstående figur ses hvordan den krypteret kommunikation mellem Kunde Serveren (Klient) og Version Server (Server), kunne have foregået:
Hvis Klient ikke allerede kender Serverens offentlige nøgle, hentes denne først. Derefter foregår alle funktionskald ved at Klienten krypterer eventuelle argumenter (som altid er xml, dvs. strenge) med serverens nøgle og derudover sender sin egen offentlige nøgle med, så eventuelt returdata kan krypteres af Serveren, inden det sendes tilbage.

9.2. Digitale Certifikater

Da en WebService stiller sine ”WebMethods” til rådighed for alle der kalder op til den, er der et stor sikkerheds aspekt som vi ikke har behandlet i projektet.
Identiteten af brugeren/processen bør bekræftes før funktionalitet der kan eksponere fortrolige information eller gør det muligt at ændre data, dette kunne gøres ved at benytte digitale certifikater.

9.3. Versions historik

Kunden har som udgangspunkt kun mulighed for at installere den seneste version af et Produkt og vil derfor kun kunne se en beskrivelse af denne version.
Hvis Kunden opgraderer med Opdaterings Komponenten fra en meget ældre version, kunne det derfor i nogle tilfælde være interessant at se hele version historikken for et Produkt.

Der skulle også være et link fra AWOConfiguratoren, hvor man kan tillade/ forhindre opgraderingen af Produkter.
En alternativ til versions historikken, er at linke til Support Sitet der er tilgængeligt på Webtops website, hvor disse oplysninger allerede er tilgængelige.

9.4. Link på påkrævede versioner

I de tilfælde hvor et Produkt ikke kan opgraderes på grund af en påkrævet version af et andet Produkt, informeres brugeren om dette i Opdatering Komponenten.
Da dette ikke nødvendigvis er et Produkt Kunden har installeret i forvejen, og Kunden derfor ikke vil se det i listen, ville det være ønskværdigt at man kunne få oplysninger om Produktet på anden vis.
Dette kunne f.eks. være i form af et link på teksten der vises i Opdatering Komponenten. Linket kunne så åbne et vindue med en beskrivelse, enten en modal som den til at følge installations processen eller et lag som i Administrations Komponenten.
Alternativt kunne dette være et link til en beskrivelse på det eksisterende supportsite.

9.5. Cirkulærer forudsætninger

Når en Webtop Administrator opretter eller redigerer Versioner i Administrations Komponenten, kan man forholdsvis let komme til at oprette cirkulære forudsætninger eller andre forudsætninger, der aldrig kan blive opfyldt. For at lette arbejdet for Webtops Administratorer, ville det være praktisk, hvis Administrations Komponenten selv undersøgte om der var sådanne umulige forudsætninger, før en Version kan gemmes.

9.6. Opgradere sig selv

Det er i den nuværende løsning ikke muligt at få AWO til at opdatere sig selv.
Problemet er, at for at opgradere et Produkt, skal alle instanser af Produktet afsluttes, da en exe/dll fil ikke kan slettes, mens den er i brug.
En mulig løsning, er at lade Opdaterings Servicen afslutte sig selv, efter at installationen er påbegyndt.

Efter den normale installation ville installationspakken så skulle genstarte servicen og fortsætte som normalt.
Det ville ikke være muligt at følge processen fra Opdaterings Komponenten, da der ikke er nogen service.
Yderligere ville servicen som udgangspunkt ”glemme” alle andre opdateringer den var i gang med, hvorfor opdateringen af den selv altid skal udføres til sidst.

KAPITEL 10

10. Konklusion

Der er implementeret et fuldt funktionsdygtigt automatisk webbaseret opdaterings-system, der gør det muligt for Webtops kunder at holde deres WebtopONE installationer opdaterede.
Løsningen anvender en algoritme til valg af opdateringerne, der gør det muligt at tage hensyn til afhængighed mellem produktversioner. Det er også muligt for systemet at håndtere at en produktversioner der kræver manuel installation og dermed ikke kan opdateres med AWO systemet.

Den algoritme der finder ud af hvilke produkter der er installeret på kundernes server og i hvilke versioner, er placeret i Opdaterings Servicen. Hvis der på et senere tidspunkt vælges et andet produkt end InstallAware til at konstuere installationspakkerne, eller der ønskes understøttelse af versioner før 5.0, er det nødvendigt at opdatere denne service. I den nuværede løsning er det ikke muligt at benytte AWO systemet til at udføre denne opdatering, som nærmere beskrevet i afsnit 9.6.

Løsningen kan, i dens nuværende form godt tages i brug, men der er dog stadig visse sikkerhedsaspekter, der bør forbedres. Dette drejer sig hovedsageligt om sikring af webservicen og de data denne håndterer.

Ved projektets afslutning var der endnu ikke en fungerende udgave af WebtopONE Coporate Portal Server 5.0. Derfor var det kun muligt at teste systemet i en begrænset, specialfremstillet udgave. Dette betyder at det heller ikke har været muligt at lave en egentlig field-test hos rigtige Webtop Kunder.

KAPITEL 11

11. Kildehenvisninger

[GEN]: Generics i .NET
http://www.microsoft.com/danmark/msdn/dothenrik/article0205.asp

[BITS]: Background Intelligent Transfer Service Reference
http://msdn.microsoft.com/library/en-us/bits/bits/bits_start_page.asp

[UAB]: Updater Application Block–Version 2.0
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnpag2/html/updaterv2.asp

[CAS]: Explore and Extend the Cassini Web Server
http://www.devx.com/dotnet/Article/11711

[RSA]: RSA Encryption in .NET
http://www.eggheadcafe.com/articles/20020630.asp

[SEC]: New and Improved Security in the .NET Framework 2.0
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnnetsec/html/foundstone.asp

[SKA]: Raport skabelon for eksamensprojekter på DTU.
http://www2.imm.dtu.dk/teaching/thesis/

[IEEE-830]: IEEE-830 standarden
Software Engineering, Principles and Practice
Hans van Vliet 2000
Forlag: Wiley
ISBN: 0-471-97508-7

KAPITEL 12

12. Bilag

12.1. Use case tests

TESTPLAN

Use case I – Opgradering af Produkter

Testplan ID.:	**version.:**	**Testtype:**	**Dato:**	**Tester:**
1	1.1	Use case test	16/1 2006	Bo S. Christensen

Testformål:
Teste funktionaliteten beskrevet i use case I – Opgradering af Produkter.

Anden information:
Da der endnu ikke findes nogen færdige komponenter til version 5.0 af webtopONE, er der blevet udviklede 4 komponenter, i 2-3 versioner med dertilhørende installere, specielt til AWO projektet.
Disse komponenter har alle deres versionsnummer stående tilhøjre for deres navn, så det er let at se om en opdatering er blevet gennemført korrekt.

Testforudsætninger:
- En tidligere version af Produktet skal allerede været installeret
- En nyere version af et installeret Produkt skal findes på Versions Serveren (opret evt. en ny version, se use case IIh)
- Versions Serveren skal køre og der skal kunne skabes forbindelse til denne.
- Opdaterings Servicen skal køre.

Udførsel:
7. Brugeren logger på sin WebtopONE portal (Kunde Serveren)
8. Brugeren vælger et eller flere Produkter som skal opgraderes og trykker opdater
9. Brugeren præsenteres for et nyt vindue hvori installationsprocessen kan følges
10. Når installationen er færdig vises en installationsoversigt.
11. Installationsoversigten lukkes og brugeren præsenteres for en opdateret liste over tilgængelig opdateringer.
12. Brugeren validere manuelt om opgraderingen er blevet fuldført, ved at tjekke versionsnummeret på komponenten.

Korrigerende handlinger ved fundne fejl:
Fejl rapporteres til den ansvarlige udvikler for delkomponenten, så denne kan blive rettet. Hvorefter testen udføres igen

TESTRAPPORT

Use case I – Opgradering af Produkter

Testrapport ID:	**version:**	**Testtype:**	**Udført:**	**Tester:**
1	1.1	Use case test	16/1 2006	Bo S. Christensen

Testresultat:

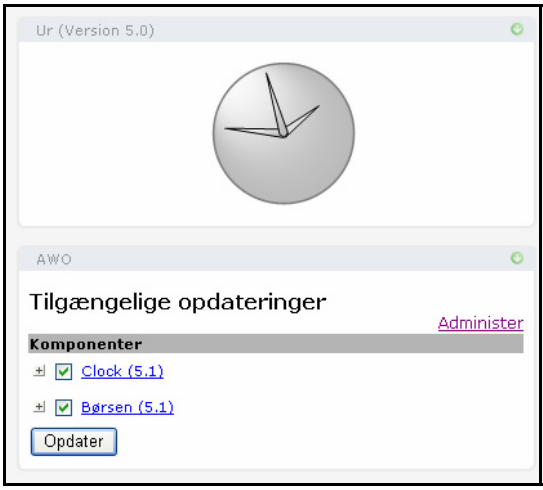


Fig 1 - Før opdatering

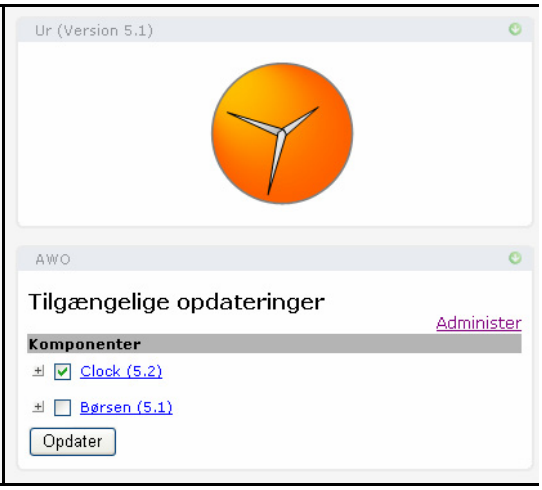


Fig 2 - Efter opdatering

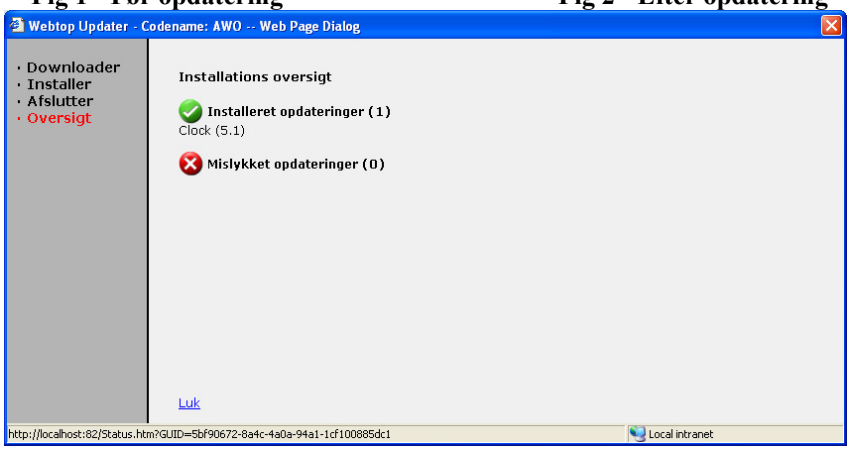


Fig 3 - Installationsoversigt

Konklusion:

På ovenstående figurer ses ”ur” komponenter før og efter opdateringen.
Som det kan ses på versionsnummerne til højre for komponentoverskriften og udfra installationsoversigten, er opdateringen gennemført korrekt

TESTPLAN

Use case Ia – Ændring af kundeoplysninger

Testplan ID.:	version.:	Testtype:	Dato:	Tester:
1a	1.0	Use case test	17/1 2006	Bo S. Christensen

Testformål:
Teste funktionaliteten beskrevet i use case Ia – Ændring af kundeoplysninger.

Anden information:
Kundeoplysninger gemmes lokalt på Kundens Server i filen AWOSettings.xml og på Versions Serveren i filen Customers.xml

Testforudsætninger:
- Versions Serveren skal køre og der skal kunne skabes forbindelse til denne.
- Opdaterings Servicen skal køre.

Udførsel:
6. Brugeren logger på sin WebtopONE portal (Kunde Serveren)
7. Brugeren klikker på ”administrer” knappen
8. Firmanavn udfylders og Produkter aktiveres/deaktivers
9. Brugeren klikker på ”ok” knappen
10. Brugeren validerer manuelt at oplysningerne er blevet gemt korrekt i de 2 filer: Customers.xml og AWOSettings.xml, på henholdsvis Versions Serveren og Kunde Serveren

Korrigerende handlinger ved fundne fejl:
Fejl rapporteres til den ansvarlige udvikler for delkomponenten, så denne kan blive rettet. Hvorefter testen udføres igen

TESTRAPPORT

Use case Ia – Ændring af kundeoplysninger

Testrapport ID:	**version:**	**Testtype:**	**Udført:**	**Tester:**
1a	1.0	Use case test	17/1 2006	Bo S. Christensen

Testresultat:

Indtastede oplysninger:

Firmanavn: DataDanmark A/S

Deaktiveret produkter: Agenda

AWOSettings.xml:

```
<Settings>
 <CompanyName>Datadanmark</CompanyName>
 <CustomerGuid>ade7b144-f67f-456c-acd4-da933c6494c3</CustomerGuid>
 <ServiceState>2</ServiceState>
 <UpdateDay>0</UpdateDay>
 <UpdateHour>13</UpdateHour>
 <DisabledProducts>
  <Product>
   <ProductName>Agenda</ProductName>
   <ProductGUID>f09f4d52-77f9-482b-9786-1538d2a1e48f</ProductGUID>
  </Product>
 </DisabledProducts>
</Settings>
```

Customers.xml:

```
<Customers>
 <Customer Name="Datadanmark" GUID="ade7b144-f67f-456c-acd4-
da933c6494c3">
  <InstalledProducts LastUpdated="16-01-2006 13:48:00">
   <VersionReference>
    <VersionNumber>1.1</VersionNumber>
    <ProductGUID>f09f4d52-77f9-482b-9786-1538d2a1e48f</ProductGUID>
   </VersionReference>
...
  </InstalledProducts>
 </Customer>
</Customers>
```

Konklusion:

Som man kan se på indholdet af de 2 filer, er Firmanavn blevet gemt korrekt på både Kundens server og på Versions Serveren. Oplysninger om det deaktiveret Produkt (Agenda) er også blevet gemt korrekt på Kunde Serveren.

TESTPLAN

Use case IIa – Kontrol af kundeoplysninger

Testplan ID.:	**version.:**	**Testtype:**	**Dato:**	**Tester:**
2a	1.0	Use case test	17/1 2006	Bo S. Christensen

Testformål:
Teste funktionaliteten beskrevet i use case IIa – Kontrol af kundeoplysninger.

Anden information:
Hver gang en kunde opdaterer et Produkt logges det på Versions Serveren i filen customers.xml og kan ses via Administrations Komponenten.

Testforudsætninger:

- Versions Serveren skal køre og der skal kunne skabes forbindelse til denne.

Udførsel:

7. Log på Kundens WebtopONE portal.
8. Noter Firmanavn og hvilke produktversioner som er installeret.
9. Log på Webtops WebtopONE portal (Webtop Server)
10. Vælg fanebladet ”kunde information”
11. VælgKkunden som blev brugt i step 1
12. Valider manuelt om data fra step 5 og 2 stemmer overens

Korrigerende handlinger ved fundne fejl:
Fejl rapporteres til den ansvarlige udvikler for delkomponenten, så denne kan blive rettet. Hvorefter testen udføres igen

TESTRAPPORT

Use case IIa – Kontrol af kundeoplysninger

Testrapport ID:	**version:**	**Testtype:**	**Udført:**	**Tester:**
2a	1.0	Use case test	17/1 2006	Bo S. Christensen

Testresultat:

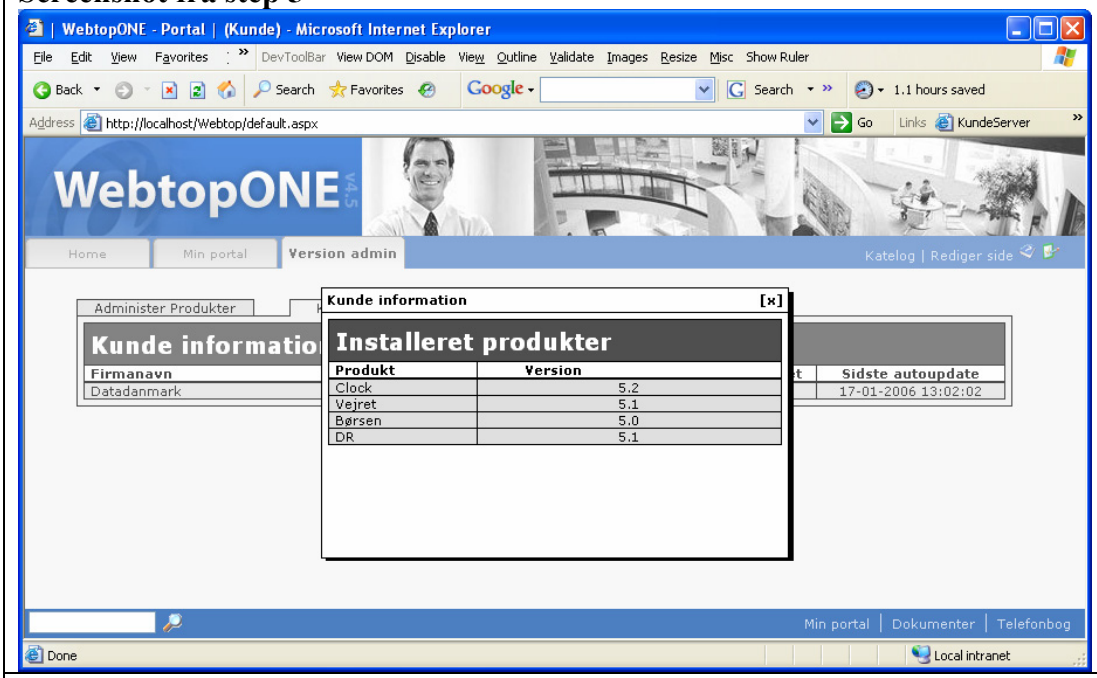
Noteret information:

Firmanavn: DataDanmark

Installede produkter:

- Clock 5.2
- Vejret 5.1
- Børsen 5.0
- DR 5.1

Screenshot fra step 5



Konklusion:

Som man kan se på screenshottet fra step 5, stemmer information om firmanavn og installeret Produkter, overens med data norteret i step 2

TESTPLAN

Use case IIb – Tilføj ny Kategori

Testplan ID.:	**version.:**	**Testtype:**	**Dato:**	**Tester:**
2b	1.0	Use case test	18/1 2006	Bo S. Christensen

Testformål:
Teste funktionaliteten beskrevet i use case IIb – Tilføj ny Kategori

Anden information:
Information om Versioner, Produkter og Kategorier bliver gemt i filen Products.xml på Versions Serveren

Testforudsætninger:
- Versions Serveren skal køre og der skal kunne skabes forbindelse til denne.

Udførsel:
5. Log på Webtops WebtopONE portal (Webtop Server)
6. Tryk på ”opret ny kategori”
7. Indtast et navn for den nye Kategori og tryk ok
8. Valider manuelt at kategorien er blevet oprettet korrekt i Products.xml

Korrigerende handlinger ved fundne fejl:
Fejl rapporteres til den ansvarlige udvikler for delkomponenten, så denne kan blive rettet. Hvorefter testen udføres igen

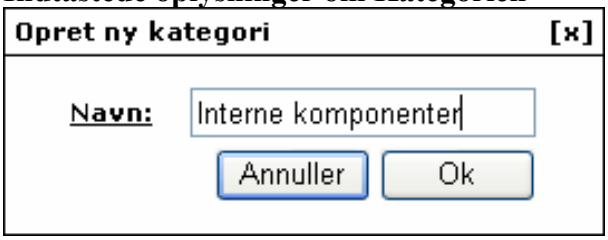
TESTRAPPORT

Use case IIb – Tilføj ny Kategori

Testrapport ID:	**version:**	**Testtype:**	**Udført:**	**Tester:**
2b	1.0	Use case test	18/1 2006	Bo S. Christensen

Testresultat:

Indtastede oplysninger om Kategorien



Udsnit fra Products.xml

```
<ProductCategories>
 ...
 <ProductCategory>
  <Name>Interne komponenter</Name>
  <GUID>bcab90ca-72d0-447b-b7ae-dedb101474ca</GUID>
  <Products />
 </ProductCategory>
</ProductCategories>
```

Konklusion:

Som man kan se fra indholdet af Products.xml, stemmer informationen overens med det indtastede.

TESTPLAN

Use case IIc – Rediger Kategori

Testplan ID.:	**version.:**	**Testtype:**	**Dato:**	**Tester:**
2c	1.0	Use case test	18/1 2006	Bo S. Christensen

Testformål:
Teste funktionaliteten beskrevet i use case IIc – Rediger Kategori

Anden information:
Information om Versioner, Produkter og Kategorier bliver gemt i filen Products.xml på Versions Serveren

Testforudsætninger:
- Versions Serveren skal køre og der skal kunne skabes forbindelse til denne.

Udførsel:
1. Log på Webtops WebtopONE portal (Webtop Server)
2. Tryk på ”rediger” ud for den Kategori som skal redigeres
3. Indtast nyt navn for Kategorien og tryk ok
4. Valider manuelt at navnet for kategorien er blevet updateret i Products.xml

Korrigerende handlinger ved fundne fejl:
Fejl rapporteres til den ansvarlige udvikler for delkomponenten, så denne kan blive rettet. Hvorefter testen udføres igen

TESTRAPPORT

Use case IIc – Rediger Kategori

Testrapport ID:	**version:**	**Testtype:**	**Udført:**	**Tester:**
2c	1.0	Use case test	18/1 2006	Bo S. Christensen

Testresultat:

Indtastede oplysninger om Kategorien

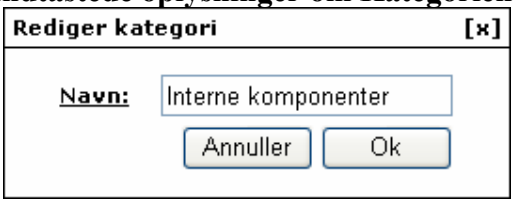


Fig 1 - Før redigering

Fig 2 - Efter redigering

Udsnit fra Products.xml før redigering

```
<ProductCategories>
 ...
 <ProductCategory>
  <Name>Interne komponenter</Name>
  <GUID>bcab90ca-72d0-447b-b7ae-dedb101474ca</GUID>
  <Products />
 </ProductCategory>
</ProductCategories>
```

Udsnit fra Products.xml efter redigering

```
<ProductCategories>
 ...
 <ProductCategory>
  <Name>Eksterne komponenter</Name>
  <GUID>bcab90ca-72d0-447b-b7ae-dedb101474ca</GUID>
  <Products />
 </ProductCategory>
</ProductCategories>
```

Konklusion:

Som man kan se fra indholdet af Products.xml efter redigeringen, er det nye navn for Kategorien blevet opdateret korrekt.

TESTPLAN

Use case IId – Slet Kategori

Testplan ID.:	version.:	Testtype:	Dato:	Tester:
2d	1.0	Use case test	18/1 2006	Bo S. Christensen

Testformål:
Teste funktionaliteten beskrevet i use case IId – Slet Kategori

Anden information:
Information om Versioner, Produkter og Kategorier bliver gemt i filen Products.xml på Versions Serveren

Testforudsætninger:
- Versions Serveren skal køre og der skal kunne skabes forbindelse til denne.

Udførsel:
1. Log på Webtops WebtopONE portal (Webtop Server)
2. Klik på ”slet” ud for den Kategori som skal slettes
3. Klik på ”ok” for at bekræfte sletningen
4. Valider manuelt at Kategorien er slettet fra Products.xml

Korrigerende handlinger ved fundne fejl:
Fejl rapporteres til den ansvarlige udvikler for delkomponenten, så denne kan blive rettet. Hvorefter testen udføres igen

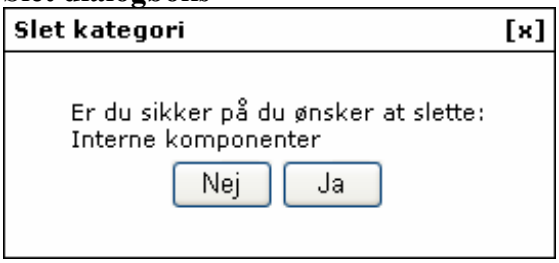
TESTRAPPORT

Use case IId – Slet Kategori

Testrapport ID:	**version:**	**Testtype:**	**Udført:**	**Tester:**
2d	1.0	Use case test	18/1 2006	Bo S. Christensen

Testresultat:

Slet dialogboks



Products.xml før kategorien slettes

```
<ProductCategories>
 <ProductCategory>
  <Name>WebtopONE</Name>
  <GUID>869e1591-6720...</GUID>
  <Products>
   ...
  </Products>
 </ProductCategory>
 <ProductCategory>
  <Name>Komponenter</Name>
  <GUID>266bcee8-3abd...</GUID>
  <Products>
   ...
  </Products>
 </ProductCategory>
 <ProductCategory>
  <Name>Interne
komponenter</Name>
  <GUID>bcab90ca-72d0...</GUID>
  <Products />
 </ProductCategory>
</ProductCategories>
```

Products.xml efter kategorien er blevet slettet

```
<ProductCategories>
 <ProductCategory>
  <Name>WebtopONE</Name>
  <GUID>869e1591-6720...</GUID>
  <Products>
   ...
  </Products>
 </ProductCategory>
 <ProductCategory>
  <Name>Komponenter</Name>
  <GUID>266bcee8-3abd...</GUID>
  <Products>
   ...
  </Products>
 </ProductCategory>
</ProductCategories>
```

Konklusion:

Som man kan se fra indholdet af Products.xml, bliver Kategorien slettet korrekt.

TESTPLAN

Use case IIe – Tilføj Produkt

Testplan ID.:	**version.:**	**Testtype:**	**Dato:**	**Tester:**
2e	1.0	Use case test	18/1 2006	Bo S. Christensen

Testformål:
Teste funktionaliteten beskrevet i use case IIe – Tilføj Produkt

Anden information:
Information om Versioner, Produkter og Kategorier bliver gemt i filen Products.xml på Versions Serveren

Testforudsætninger:
- Versions Serveren skal køre og der skal kunne skabes forbindelse til denne.

Udførsel:
1. Log på Webtops WebtopONE portal (Webtop Server)
2. Tryk på ”opret nyt produkt” under den Kategori hvortil Produktet skal tilføjes.
3. Udfyld Navn, beskrivelse og Produktets GUID og Klik ”ok”
4. Valider manuelt at produktet er blevet oprettet korrekt i Products.xml

Korrigerende handlinger ved fundne fejl:
Fejl rapporteres til den ansvarlige udvikler for delkomponenten, så denne kan blive rettet. Hvorefter testen udføres igen

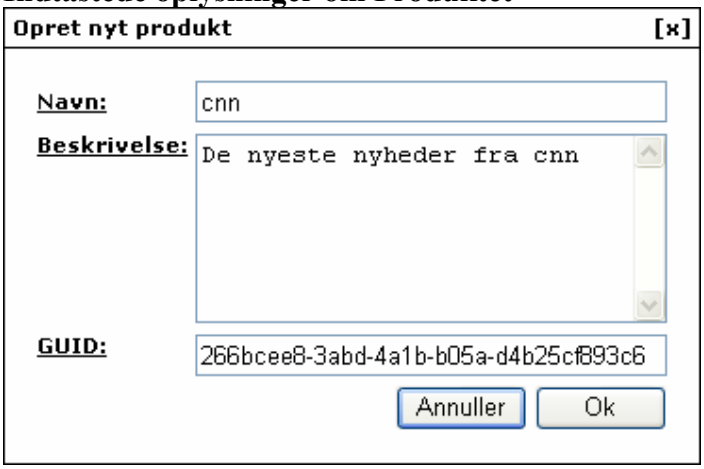
TESTRAPPORT

Use case IIe– Tilføj Produkt

Testrapport ID:	**version:**	**Testtype:**	**Udført:**	**Tester:**
2e	1.0	Use case test	18/1 2006	Bo S. Christensen

Testresultat:

Indtastede oplysninger om Produktet



Udsnit fra Products.xml

```
<ProductCategories>
 <ProductCategory>
  <Name>WebtopONE</Name>
  <GUID>869e1591-6720-4bea-a034-e2daddb95fff</GUID>
  <Products>
   <Product>
    <Name>cnn</Name>
    <GUID>266bcee8-3abd-4a1b-b05a-d4b25cf893c6</GUID>
    <Description>De nyeste nyheder fra cnn</Description>
    <Versions />
   </Product>
   ...
  </Products>
 </ProductCategory>
  ...
</ProductCategories>
```

Konklusion:

Som man kan se fra indholdet af Products.xml, stemmer informationen overens med det indtastede.

TESTPLAN

Use case IIf – Rediger Produkt

Testplan ID.:	**version.:**	**Testtype:**	**Dato:**	**Tester:**
2f	1.0	Use case test	18/1 2006	Bo S. Christensen

Testformål:
Teste funktionaliteten beskrevet i use case IIf – Rediger Produkt

Anden information:
Information om Versioner, Produkter og Kategorier bliver gemt i filen Products.xml på Versions Serveren

Testforudsætninger:
- Versions Serveren skal køre og der skal kunne skabes forbindelse til denne.

Udførsel:
1. Log på Webtops WebtopONE portal (Webtop Server)
2. Tryk på ”rediger” ud for det Produkt som skal redigeres
3. Opdater oplysningerne for Produktet.
4. Tryk ”ok”
5. Valider manuelt at oplysningerne om produktet er blevet opdateret i Products.xml

Korrigerende handlinger ved fundne fejl:
Fejl rapporteres til den ansvarlige udvikler for delkomponenten, så denne kan blive rettet. Hvorefter testen udføres igen

TESTRAPPORT

Use case IIf– Rediger Produkt

Testrapport ID:	**version:**	**Testtype:**	**Udført:**	**Tester:**
2f	1.0	Use case test	18/1 2006	Bo S. Christensen

Testresultat:

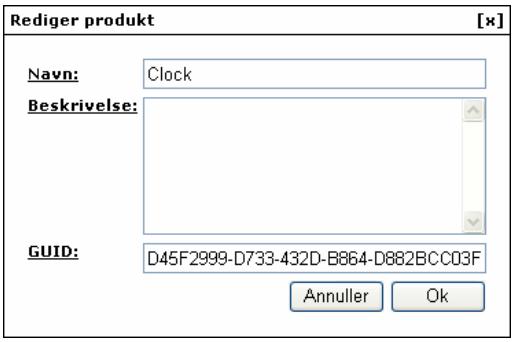


Fig 1 – Før redigering

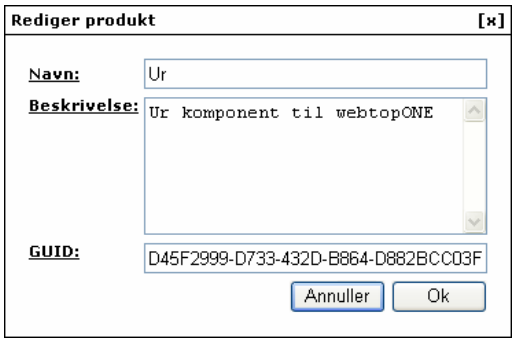


Fig 2 – Efter redigering

Udsnit fra Products.xml før redigering

```
<ProductCategories>
  <ProductCategory>
    <Name>Komponenter</Name>
    <GUID>266bcee8-3abd-4a1b-
b05a-d4b25cf893c6</GUID>
    <Products>
      <Product>
        <Name>Clock</Name>
        <GUID>D45F2999-D733-
432D-B864-D882BCC03FE6</GUID>
        <Description />
        <Versions />
      </Product>
    </Products>
  </ProductCategory>
</ProductCategories>
```

Udsnit fra Products.xml efter redigering

```
<ProductCategories>
 <ProductCategory>
  <Name>Komponenter</Name>
  <GUID>266bcee8-3abd-4a1b-
b05a-d4b25cf893c6</GUID>
  <Products>
   <Product>
    <Name>Ur</Name>
    <GUID>D45F2999-D733-432D-
B864-D882BCC03FE6</GUID>
    <Description>Ur komponent
til webtopONE<Description>
    <Versions />
   </Product>
  </Products>
 </ProductCategory>
</ProductCategories>
```

Konklusion:

Som man kan se fra indholdet af Products.xml efter redigering, er de ændret data (Navn og beskrivelse) blevet opdateret korrekt.

TESTPLAN

Use case IIg – Slet Produkt

Testplan ID.:	**version.:**	**Testtype:**	**Dato:**	**Tester:**
2g	1.0	Use case test	18/1 2006	Bo S. Christensen

Testformål:
Teste funktionaliteten beskrevet i use case IIg – Slet Produkt

Anden information:
Information om Versioner, Produkter og Kategorier bliver gemt i filen Products.xml på Versions Serveren

Testforudsætninger:

- Versions Serveren skal køre og der skal kunne skabes forbindelse til denne.

Udførsel:

1. Log på Webtops WebtopONE portal (Webtop Server)
2. Klik på ”slet” ud for det Produkt som skal slettes
3. Klik på ”ok” for at bekræfte sletningen
4. Valider manuelt at Produktet er slettet fra Producs.xml

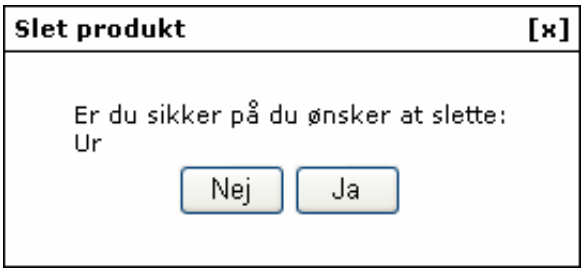
Korrigerende handlinger ved fundne fejl:
Fejl rapporteres til den ansvarlige udvikler for delkomponenten, så denne kan blive rettet. Hvorefter testen udføres igen

TESTRAPPORT

Use case IIg– Slet Produkt

Testrapport ID:	**version:**	**Testtype:**	**Udført:**	**Tester:**
2g	1.0	Use case test	18/1 2006	Bo S. Christensen

Testresultat:



Products.xml før produktet slettes

```
<ProductCategories>
 ...
 <ProductCategory>
  <Name>Komponenter</Name>
  <GUID>266bcee8-3abd...</GUID>
  <Products>
   <Product>
    <Name>Ur</Name>
    <GUID>D45F2999-D7...</GUID>
    <Description />
    <Versions />
   </Product>
   <Product>
    <Name>Børsen</Name>
    <GUID>5EDB62A9-00...</GUID>
    <Description />
    <Versions />
   </Product>
  </Products>
 </ProductCategory>
</ProductCategories>
```

Products.xml efter produktet er blevet slettet

```
<ProductCategories>
 ...
 <ProductCategory>
  <Name>Komponenter</Name>
  <GUID>266bcee8-3abd...</GUID>
  <Products>
   <Product>
    <Name>Børsen</Name>
    <GUID>5EDB62A9-00...</GUID>
    <Description />
    <Versions />
   </Product>
  </Products>
 </ProductCategory>
</ProductCategories>
```

Konklusion:
Som man kan se fra indholdet af Products.xml, bliver Produktet slettet korrekt

TESTPLAN

Use case IIh – Tilføj ny Version

Testplan ID.:	**version.:**	**Testtype:**	**Dato:**	**Tester:**
2h	1.0	Use case test	18/1 2006	Bo S. Christensen

Testformål:
Teste funktionaliteten beskrevet i use case IIh – Tilføj ny Version.

Anden information:
Information om Versioner, Produkter og Kategorier bliver gemt i filen Products.xml på Versions Serveren

Testforudsætninger:
- Versions Serveren skal køre og der skal kunne skabes forbindelse til denne.

Udførsel:
7. Log på Webtops WebtopONE portal (Webtop Server)
8. Vælg et Produkt på listen eller opret et nyt (se usecase IIe)
9. Klik på versioner udfor den valgte produkt.
10. Indtast oplysninger om den nye Version, tilføj installationsfil og eventuelle installationskrav.
11. Tryk ok og tjeck om den nye version bliver tilføjet til versionslisten
12. Valider at versionsinformationerne er gemt korrekt i Products.xml

Korrigerende handlinger ved fundne fejl:
Fejl rapporteres til den ansvarlige udvikler for delkomponenten, så denne kan blive rettet. Hvorefter testen udføres igen

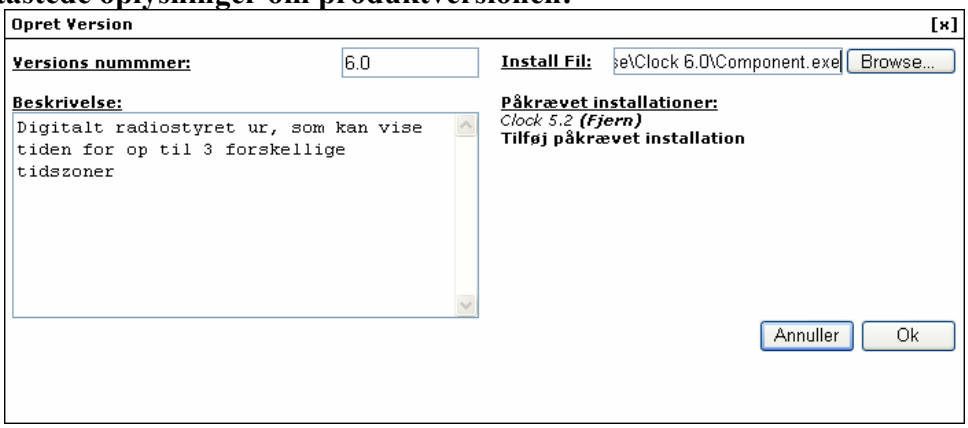
TESTRAPPORT

Use case IIh – Tilføj ny Version

Testrapport ID:	version:	Testtype:	Udført:	Tester:
2h	1.0	Use case test	18/1 2006	Bo S. Christensen

Testresultat:

Indtastede oplysninger om produktversionen:



Udsnit fra Products.xml

```
<Version>
 <VersionNumber>6.0</VersionNumber>
 <Active>True</Active>
 <GUID>31cadad9-a90e-4bd7-8d9e-28b44bf18c76</GUID>
 <Description>Digitalt radiostyret ur, som kan vise tiden for op
til 3 forskellige tidszoner</Description>
 <FilePath>http://localhost/version/data/ 6.0-
Component.exe</FilePath>
 <FileSize>1028513</FileSize>
 <CommandArgument>/s</CommandArgument>
 <RequiredVersions>
  <VersionReference>
   <VersionNumber>5.2</VersionNumber>
   <ProductGUID>D45F2999-D733-432D-B864-D882BCC03FE6</ProductGUID>
  </VersionReference>
 </RequiredVersions>
</Version>
```

Konklusion:

Som man kan se fra indholdet af Products.xml, stemmer informationen overens med det indtastede.

TESTPLAN

Use case IIi – Rediger Version

Testplan ID.:	**version.:**	**Testtype:**	**Dato:**	**Tester:**
2i	1.0	Use case test	18/1 2006	Bo S. Christensen

Testformål:
Teste funktionaliteten beskrevet i use case IIi – Rediger Version

Anden information:
Information om Versioner, Produkter og Kategorier bliver gemt i filen Products.xml på Versions Serveren

Testforudsætninger:
- Versions Serveren skal køre og der skal kunne skabes forbindelse til denne.

Udførsel:
1. Log på Webtops WebtopONE portal (Webtop Server)
2. Vælg Produkt
3. Tryk på ”Rediger” ud for den Version som skal redigeres
4. Opdater oplysninger og upload eventuel ny installpakke for Versionen
5. Tryk ”ok”
6. Valider manuelt af oplysningerne om Versionen er blevet opdateret i products.xml

Korrigerende handlinger ved fundne fejl:
Fejl rapporteres til den ansvarlige udvikler for delkomponenten, så denne kan blive rettet. Hvorefter testen udføres igen

TESTRAPPORT

Use case IIi– Rediger Version

Testrapport ID:	**version:**	**Testtype:**	**Udført:**	**Tester:**
2i	1.0	Use case test	18/1 2006	Bo S. Christensen

Testresultat:

Versionsoplysninger før opdatering	**Versionsoplysninger efter opdatering**
Versionsnummer: 5.2 **Beskrivelse:** Analog ur som anvender klientens tidsinstillinger	**Versionsnummer:** 5.1.1 **Beskrivelse:** Digitalt ur som anvender klientens tidsinstillinger

Udsnit af Products.xml før opdatering af version

```
<Product>
 <Name>Clock</Name>
 <GUID>D45F2999-D733-...</GUID>
 <Description>...</Description>
 <Versions>
 ...
  <Version>
   <VerNumber>5.2</VerNumber>
   <Active>True</Active>
   <GUID>6e776049-cca...</GUID>
   <Description>
Analog ur som anvender klients
tidsindstillinger
   </Description>
   ...
  </Version>
 </Versions>
</Product>
```

Udsnit af Products.xml efter opdatering af version

```
<Product>
 <Name>Clock</Name>
 <GUID>D45F2999-D733-...</GUID>
 <Description>...</Description>
 <Versions>
 ...
  <Version>
   <VerNumber>5.1.1</VerNumber>
   <Active>True</Active>
   <GUID>6e776049-cca...</GUID>
   <Description>
Digitalt ur som anvender
klientens tidsinstillinger
</Description>
   ...
  </Version>
 </Versions>
</Product>
```

Konklusion:
Som man kan se fra indholdet af Products.xml, stemmer informationen overens med det indtastede.

TESTPLAN

Use case IIj – Slet Version

Testplan ID.:	version.:	Testtype:	Dato:	Tester:
2j	1.0	Use case test	18/1 2006	Bo S. Christensen

Testformål:
Teste funktionaliteten beskrevet i use case IIj – Slet Version

Anden information:
Information om Versioner, Produkter og Kategorier bliver gemt i filen Products.xml på Versions Serveren

Testforudsætninger:
- Versions Serveren skal køre og der skal kunne skabes forbindelse til denne.

Udførsel:
1. Log på Webtops WebtopONE portal (Webtop Server)
2. Vælg Produkt
3. Klik på ”slet” ud for den Version som skal slettes
4. Klik på ”ok” for at bekræfte sletningen
5. Valider manuelt at Versionen er slettet fra Products.xml

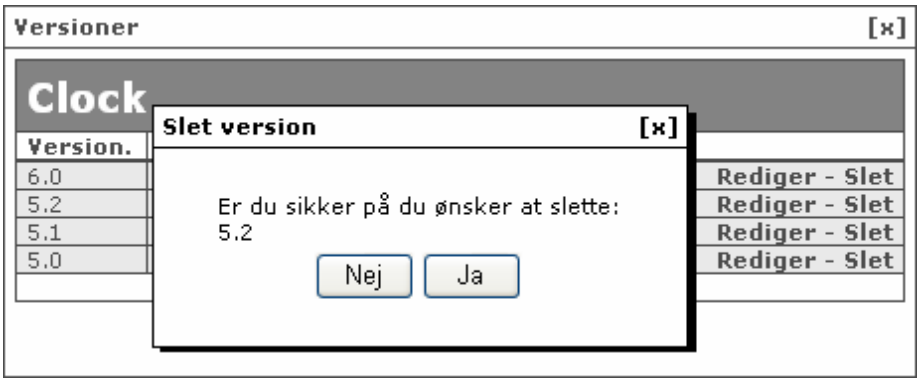
Korrigerende handlinger ved fundne fejl:
Fejl rapporteres til den ansvarlige udvikler for delkomponenten, så denne kan blive rettet. Hvorefter testen udføres igen

TESTRAPPORT

Use case IIj– Slet Version

Testrapport ID:	version:	Testtype:	Udført:	Tester:
2j	1.0	Use case test	18/1 2006	Bo S. Christensen

Testresultat:



Products.xml før sletning

```
<Product>
 <Name>Clock</Name>
 <GUID>D45F2999-D733-...</GUID>
 <Description>...</Description>
 <Versions>
  <Version>
   <VerNumber>6.0</VerNumber>
   ...
  </Version>
  <Version>
   <VerNumber>5.2</VerNumber>
   ...
  </Version>
  <Version>
   <VerNumber>5.1</VerNumber>
   ...
  </Version>
  ...
 </Versions>
</Product>
```

Products.xml efter sletning

```
<Product>
 <Name>Clock</Name>
 <GUID>D45F2999-D733-...</GUID>
 <Description>...</Description>
 <Versions>
  <Version>
   <VerNumber>6.0</VerNumber>
   ...
  </Version>
  <Version>
   <VerNumber>5.1</VerNumber>
   ...
  </Version>
  ...
 </Versions>
</Product>
```

Konklusion:

Som man kan se fra indholdet af Products.xml, bliver Versionen slettet korrekt

12.2. Kildekode

BITS

InteropBits.cs

```
using System;
using System.Runtime.InteropServices;

namespace BITS.Interop
{
    [GuidAttribute("4991D34B-80A1-4291-83B6-3328366B9097")]
    [ClassInterfaceAttribute(ClassInterfaceType.None)]
    [ComImportAttribute()]
    public class BackgroundCopyManager { }

    [InterfaceTypeAttribute(ComInterfaceType.InterfaceIsIUnknown)]
    [GuidAttribute("5CE34C0D-0DC9-4C1F-897C-DAA1B78CEE7C")]
    [ComImportAttribute()]
    public interface IBackgroundCopyManager
    {
        void CreateJob([MarshalAs(UnmanagedType.LPWStr)] string DisplayName, BG_JOB_TYPE Type,
out Guid pJobId, [MarshalAs(UnmanagedType.Interface)] out IBackgroundCopyJob ppJob);
        void GetJob(ref Guid jobID, [MarshalAs(UnmanagedType.Interface)] out IBackgroundCopyJob
ppJob);
        void EnumJobs(uint dwFlags, [MarshalAs(UnmanagedType.Interface)] out
IEnumBackgroundCopyJobs ppenum);
        void GetErrorDescription([MarshalAs(UnmanagedType.Error)] int hResult, uint LanguageId,
[MarshalAs(UnmanagedType.LPWStr)] out string pErrorDescription);
    }

    [GuidAttribute("97EA99C7-0186-4AD4-8DF9-C5B4E0ED6B22")]
    [InterfaceTypeAttribute(ComInterfaceType.InterfaceIsIUnknown)]
    [ComImportAttribute()]
    public interface IBackgroundCopyCallback
    {
        void JobTransferred([MarshalAs(UnmanagedType.Interface)] IBackgroundCopyJob pJob);
        void JobError([MarshalAs(UnmanagedType.Interface)] IBackgroundCopyJob pJob,
[MarshalAs(UnmanagedType.Interface)] IBackgroundCopyError pError);
        void JobModification([MarshalAs(UnmanagedType.Interface)] IBackgroundCopyJob pJob, uint
dwReserved);
    }

    [GuidAttribute("19C613A0-FCB8-4F28-81AE-897C3D078F81")]
    [InterfaceTypeAttribute(ComInterfaceType.InterfaceIsIUnknown)]
    [ComImportAttribute()]
    public interface IBackgroundCopyError
    {
        void GetError(out BG_ERROR_CONTEXT pContext, [MarshalAs(UnmanagedType.Error)] out int
pCode);
        void GetFile([MarshalAs(UnmanagedType.Interface)] out IBackgroundCopyFile pVal);
        void GetErrorDescription(uint LanguageId, [MarshalAs(UnmanagedType.LPWStr)] out string
pErrorDescription);
        void GetErrorContextDescription(uint LanguageId, [MarshalAs(UnmanagedType.LPWStr)] out
string pContextDescription);
        void GetProtocol([MarshalAs(UnmanagedType.LPWStr)] out string pProtocol);
    }

    [GuidAttribute("01B7BD23-FB88-4A77-8490-5891D3E4653A")]
    [InterfaceTypeAttribute(ComInterfaceType.InterfaceIsIUnknown)]
    [ComImportAttribute()]
    public interface IBackgroundCopyFile
    {
        void GetRemoteName([MarshalAs(UnmanagedType.LPWStr)] out string pVal);
        void GetLocalName([MarshalAs(UnmanagedType.LPWStr)] out string pVal);
        void GetProgress(out _BG_FILE_PROGRESS pVal);
    }

    [InterfaceTypeAttribute(ComInterfaceType.InterfaceIsIUnknown)]
```

```
    [GuidAttribute("37668D37-507E-4160-9316-26306D150B12")]
    [ComImportAttribute()]
    public interface IBackgroundCopyJob
    {
        void AddFileSet(uint cFileCount, ref _BG_FILE_INFO pFileSet);
        void AddFile([MarshalAs(UnmanagedType.LPWStr)] string RemoteUrl,
[MarshalAs(UnmanagedType.LPWStr)] string LocalName);
        void EnumFiles([MarshalAs(UnmanagedType.Interface)] out IEnumBackgroundCopyFiles
pEnum);
        void Suspend();
        void Resume();
        void Cancel();
        void Complete();
        void GetId(out Guid pVal);
        void GetType(out BG_JOB_TYPE pVal);
        void GetProgress(out _BG_JOB_PROGRESS pVal);
        void GetTimes(out _BG_JOB_TIMES pVal);
        void GetState(out BG_JOB_STATE pVal);
        void GetError([MarshalAs(UnmanagedType.Interface)] out IBackgroundCopyError ppError);
        void GetOwner([MarshalAs(UnmanagedType.LPWStr)] out string pVal);
        void SetDisplayName([MarshalAs(UnmanagedType.LPWStr)] string Val);
        void GetDisplayName([MarshalAs(UnmanagedType.LPWStr)] out string pVal);
        void SetDescription([MarshalAs(UnmanagedType.LPWStr)] string Val);
        void GetDescription([MarshalAs(UnmanagedType.LPWStr)] out string pVal);
        void SetPriority(BG_JOB_PRIORITY Val);
        void GetPriority(out BG_JOB_PRIORITY pVal);
        void SetNotifyFlags(BG_JOB_NOTIFICATION_TYPE Val);
        //void SetNotifyFlags(uint Val);
        void GetNotifyFlags(out uint pVal);
        void SetNotifyInterface([MarshalAs(UnmanagedType.IUnknown)] object Val);
        void GetNotifyInterface([MarshalAs(UnmanagedType.IUnknown)] out object pVal);
        void SetMinimumRetryDelay(uint Seconds);
        void GetMinimumRetryDelay(out uint Seconds);
        void SetNoProgressTimeout(uint Seconds);
        void GetNoProgressTimeout(out uint Seconds);
        void GetErrorCount(out uint Errors);
        void SetProxySettings(BG_JOB_PROXY_USAGE ProxyUsage, [MarshalAs(UnmanagedType.LPWStr)]
string ProxyList, [MarshalAs(UnmanagedType.LPWStr)] string ProxyBypassList);
        void GetProxySettings(out BG_JOB_PROXY_USAGE pProxyUsage,
[MarshalAs(UnmanagedType.LPWStr)] out string pProxyList, [MarshalAs(UnmanagedType.LPWStr)] out
string pProxyBypassList);
        void TakeOwnership();
    }

    [InterfaceTypeAttribute(ComInterfaceType.InterfaceIsIUnknown)]
    [GuidAttribute("CA51E165-C365-424C-8D41-24AAA4FF3C40")]
    [ComImportAttribute()]
    public interface IEnumBackgroundCopyFiles
    {
        void Next(uint celt, [MarshalAs(UnmanagedType.Interface)] out IBackgroundCopyFile
rgelt, out uint pceltFetched);
        void Skip(uint celt);
        void Reset();
        void Clone([MarshalAs(UnmanagedType.Interface)] out IEnumBackgroundCopyFiles ppenum);
        void GetCount(out uint puCount);
    }

    [InterfaceTypeAttribute(ComInterfaceType.InterfaceIsIUnknown)]
    [GuidAttribute("1AF4F612-3B71-466F-8F58-7B6F73AC57AD")]
    [ComImportAttribute()]
    public interface IEnumBackgroundCopyJobs
    {
        void Next(uint celt, [MarshalAs(UnmanagedType.Interface)] out IBackgroundCopyJob rgelt,
out uint pceltFetched);
        void Skip(uint celt);
        void Reset();
        void Clone([MarshalAs(UnmanagedType.Interface)] out IEnumBackgroundCopyJobs ppenum);
        void GetCount(out uint puCount);
    }
```

```
public enum BG_ERROR_CONTEXT
{
    BG_ERROR_CONTEXT_NONE = 0,
    BG_ERROR_CONTEXT_UNKNOWN = 1,
    BG_ERROR_CONTEXT_GENERAL_QUEUE_MANAGER = 2,
    BG_ERROR_CONTEXT_QUEUE_MANAGER_NOTIFICATION = 3,
    BG_ERROR_CONTEXT_LOCAL_FILE = 4,
    BG_ERROR_CONTEXT_REMOTE_FILE = 5,
    BG_ERROR_CONTEXT_GENERAL_TRANSPORT = 6,
}

public enum BG_JOB_PRIORITY
{
    BG_JOB_PRIORITY_FOREGROUND = 0,
    BG_JOB_PRIORITY_HIGH = 1,
    BG_JOB_PRIORITY_NORMAL = 2,
    BG_JOB_PRIORITY_LOW = 3,
}

public enum BG_JOB_PROXY_USAGE
{
    BG_JOB_PROXY_USAGE_PRECONFIG = 0,
    BG_JOB_PROXY_USAGE_NO_PROXY = 1,
    BG_JOB_PROXY_USAGE_OVERRIDE = 2,
}

public enum BG_JOB_STATE
{
    BG_JOB_STATE_QUEUED = 0,
    BG_JOB_STATE_CONNECTING = 1,
    BG_JOB_STATE_TRANSFERRING = 2,
    BG_JOB_STATE_SUSPENDED = 3,
    BG_JOB_STATE_ERROR = 4,
    BG_JOB_STATE_TRANSIENT_ERROR = 5,
    BG_JOB_STATE_TRANSFERRED = 6,
    BG_JOB_STATE_ACKNOWLEDGED = 7,
    BG_JOB_STATE_CANCELLED = 8,
}

public enum BG_JOB_TYPE
{
    BG_JOB_TYPE_DOWNLOAD = 0,
}

[StructLayoutAttribute(LayoutKind.Sequential, Pack = 4, Size = 0)]
public struct _BG_FILE_INFO
{
    [MarshalAs(UnmanagedType.LPWStr)]
    public string RemoteName;

    [MarshalAs(UnmanagedType.LPWStr)]
    public string LocalName;
}

[StructLayoutAttribute(LayoutKind.Sequential, Pack = 8, Size = 0)]
public struct _BG_FILE_PROGRESS
{
    public ulong BytesTotal;
    public ulong BytesTransferred;
    public int Completed;
}

[StructLayoutAttribute(LayoutKind.Sequential, Pack = 8, Size = 0)]
public struct _BG_JOB_PROGRESS
{
    public ulong BytesTotal;
    public ulong BytesTransferred;
    public uint FilesTotal;
    public uint FilesTransferred;
}
```

```
    [StructLayoutAttribute(LayoutKind.Sequential, Pack = 4, Size = 0)]
    public struct _BG_JOB_TIMES
    {
        public _FILETIME CreationTime;
        public _FILETIME ModificationTime;
        public _FILETIME TransferCompletionTime;
    }

    [StructLayoutAttribute(LayoutKind.Sequential, Pack = 4, Size = 0)]
    public struct _FILETIME
    {
        public uint dwLowDateTime;
        public uint dwHighDateTime;
    }
    [Flags]
    public enum BG_JOB_NOTIFICATION_TYPE : uint
    {
        BG_NOTIFY_JOB_TRANSFERRED = 0x0001,
        BG_NOTIFY_JOB_ERROR = 0x0002,
        BG_NOTIFY_DISABLE = 0x0004,
        BG_NOTIFY_JOB_MODIFICATION = 0x0008,
    }

    // http://msdn.microsoft.com/library/default.asp?url=/library/en-
us/bits/bits/bg_error_context.asp
    public enum ErrorCodes : uint
    {
        BG_S_PARTIAL_COMPLETE = 0x00200017,
        BG_S_UNABLE_TO_DELETE_FILES = 0x0020001A,
        BG_E_NOT_FOUND = 0x80200001,
        BG_E_INVALID_STATE = 0x80200002,
        BG_E_EMPTY = 0x80200003,
        BG_E_FILE_NOT_AVAILABLE = 0x80200004,
        BG_E_PROTOCOL_NOT_AVAILABLE = 0x80200005,
        BG_E_DESTINATION_LOCKED = 0x8020000D,
        BG_E_VOLUME_CHANGED = 0x8020000E,
        BG_E_ERROR_INFORMATION_UNAVAILABLE = 0x8020000F,
        BG_E_NETWORK_DISCONNECTED = 0x80200010,
        BG_E_MISSING_FILE_SIZE = 0x80200011,
        BG_E_INSUFFICIENT_HTTP_SUPPORT = 0x80200012,
        BG_E_INSUFFICIENT_RANGE_SUPPORT = 0x80200013,
        BG_E_REMOTE_NOT_SUPPORTED = 0x80200014,
        BG_E_NEW_OWNER_DIFF_MAPPING = 0x80200015,
        BG_E_NEW_OWNER_NO_FILE_ACCESS = 0x80200016,
        BG_E_PROXY_LIST_TOO_LARGE = 0x80200018,
        BG_E_PROXY_BYPASS_LIST_TOO_LARGE = 0x80200019,
        BG_E_TOO_MANY_FILES = 0x8020001C,
        BG_E_LOCAL_FILE_CHANGED = 0x8020001D,
        BG_E_TOO_LARGE = 0x80200020,
        BG_E_STRING_TOO_LONG = 0x80200021,
        BG_E_CLIENT_SERVER_PROTOCOL_MISMATCH = 0x80200022,
        BG_E_SERVER_EXECUTE_ENABLED = 0x80200023,
        BG_E_USERNAME_TOO_LARGE = 0x80200025,
        BG_E_PASSWORD_TOO_LARGE = 0x80200026,
        BG_E_INVALID_AUTH_TARGET = 0x80200027,
        BG_E_INVALID_AUTH_SCHEME = 0x80200028,
        BG_E_INVALID_RANGE = 0x8020002B,
        BG_E_OVERLAPPING_RANGES = 0x8020002C,
        BG_E_BLOCKED_BY_POLICY = 0x8020003E,
        BG_E_INVALID_PROXY_INFO = 0x8020003F,
        BG_E_HTTP_ERROR_400 = 0x80190190,
        BG_E_HTTP_ERROR_401 = 0x80190191,
        BG_E_HTTP_ERROR_404 = 0x80190194,
        BG_E_HTTP_ERROR_407 = 0x80190197,
        BG_E_HTTP_ERROR_414 = 0x8019019E,
        BG_E_HTTP_ERROR_501 = 0x801901F5,
        BG_E_HTTP_ERROR_503 = 0x801901F7,
        BG_E_HTTP_ERROR_504 = 0x801901F8,
        BG_E_HTTP_ERROR_505 = 0x801901F9
```

```
    }
}
```

BITSdownloader.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Net;
using System.Net.Sockets;
using BITS.Interop;

namespace BITS
{
    public class BITSdownloader
    {
        // ### Public properties
        public enum DownloadState : int
        {
            Undefinded = -1,
            Queued = 0,
            Connecting = 1,
            Transferring = 2,
            Suspended = 3,
            Error = 4,
            TransientError = 5,
            Transferred = 6,
            Acknowledged = 7,
            Cancelled = 8
        }

        public delegate void ProgressDelegate(BITSdownloader Sender, ulong BytesTransferred,
ulong BytesTotal);
        public delegate void DownloadStateChangedDelegate(BITSdownloader Sender, DownloadState
State);
        public delegate void EmptyDelegate(BITSdownloader Sender);
        public delegate void ErrorDelegate(BITSdownloader Sender, BitsError Err);

        public event ProgressDelegate ProgressChanged;
        public event DownloadStateChangedDelegate DownloadStateChanged;
        public event EmptyDelegate DownloadComplete;
        public event ErrorDelegate Error;

        public DownloadState State
        {
            get { return _State; }
        }
        public double Progress
        {
            get { return _BytesTotal > 0 ? 100 * (double)_BytesTransferred /
(double)_BytesTotal : 0; }
        }
        public ulong BytesTransferred
        {
            get
            {
                return _BytesTransferred;
            }
        }
        public ulong BytesTotal
        {
            get { return _BytesTotal; }
        }

        // ### Public class
        public class BitsError
        {
            // ### Private properties
```

```
            private IBackgroundCopyError Error;

            // ### Public properties
            public string HexCode
            {
                get
                {
                    int Code = 0;
                    BG_ERROR_CONTEXT ErrorContext;
                    Error.GetError(out ErrorContext, out Code);
                    return Byte2Hex(BitConverter.GetBytes(Code));
                }
            }
            public uint Code
            {
                get
                {
                    int Code = 0;
                    BG_ERROR_CONTEXT ErrorContext;
                    Error.GetError(out ErrorContext, out Code);
                    return ErrorInt2Uint(Code);
                }
            }
            public string Description
            {
                get
                {
                    switch (Code)
                    {
                        case 0x00200017: return "A subset of the job's files transferred
successfully before the IBackgroundCopyJob::Complete method was called. Those that were not
complete were deleted.";
                        case 0x0020001A: return "Unable to delete all temporary files
associated with the job.";
                        case 0x80200001: return "The requested job was not found.";
                        case 0x80200002: return "The requested action is not allowed in the
current job state.";
                        case 0x80200003: return "The job must contain one or more files before
you can resume the job.";
                        case 0x80200004: return "File information is not available because the
error is not associated with a local or remote file.";
                        case 0x80200005: return "Protocol information is not available because
the error is not associated with the specified transfer protocol.";
                        case 0x8020000D: return "The destination file system volume, specified
in the local file name, is locked.";
                        case 0x8020000E: return "The destination volume, specified in the local
file name, has changed. For example, the original floppy disk has been replaced with a
different floppy disk.";
                        case 0x8020000F: return "Error information is only available when the
state of the job is BG_JOB_STATE_ERROR. The error information is not available after BITS
begins transferring the job's data or the client exits.";
                        case 0x80200010: return "The network card is inactive or disconnected.
All jobs are placed in the BG_JOB_STATE_TRANSIENT_ERROR state.";
                        case 0x80200011: return "The server did not return the file size. BITS
only transfers static content and requires the HTTP server to return the Content-Length header.
The transfer request fails if the URL points to dynamic content.";
                        case 0x80200012: return "The server does not support the HTTP/1.1
protocol.";
                        case 0x80200013: return "The server does not support the Content-Range
header. Typically, you receive this error when you try to download dynamic content. You can
also receive this error if an intermediate proxy is removing the Content-Range or Content-
Length header.";
                        case 0x80200014: return "Remote use of BITS is not supported. For more
information, see Users and Network Connections.";
                        case 0x80200015: return "The network drive mapping for the local file
is different for the current owner than for the previous owner.";
                        case 0x80200016: return "The new owner has insufficient access
privileges to the temporary job files.";
                        case 0x80200018: return "The HTTP proxy list is too long. The list must
not exceed 32 KB.";
```

```
                        case 0x80200019: return "The HTTP proxy bypass list is too long. The
list must not exceed 32 KB.";
                        case 0x8020001C: return "You cannot add more than one file to an upload
job.";
                        case 0x8020001D: return "The contents of the local file changed after
the transfer process began. The contents of the local file cannot change after the transfer
process begins on an upload or upload-reply job.";
                        case 0x80200020: return "The size of the upload file exceeds the
maximum allowed upload size specified on the server.";
                        case 0x80200021: return "The specified string is too long.";
                        case 0x80200022: return "The client and server were unable to negotiate
a protocol to use for the upload job.";
                        case 0x80200023: return "Scripting or execute permissions are enabled
on the IIS virtual directory associated with the job. To upload files to the virtual directory,
disable the scripting and execute permissions on the virtual directory.";
                        case 0x80200025: return "The user name cannot exceed 300 characters.";
                        case 0x80200026: return "The password cannot exceed 300 characters.";
                        case 0x80200027: return "The specified authentication target is not
valid.";
                        case 0x80200028: return "The specified authentication scheme is not
valid.";
                        case 0x8020002B: return "The specified byte range is invalid. The byte
range must exist within the specified remote file.";
                        case 0x8020002C: return "The list of byte ranges contains overlapping
or duplicate ranges, which are not supported.";
                        case 0x8020003E: return "Group Policy settings prevent background jobs
from running at this time. For details, see the MaxInternetBandwidth policy.";
                        case 0x8020003F: return "Run-time error that indicates the proxy list
or proxy bypass list that you specified using the IBackgroundCopyJob::SetProxySettings method
is invalid.";
                        case 0x80190190: return "The server could not process the transfer
request because the syntax of the remote file name is invalid.";
                        case 0x80190191: return "The user does not have permission to access
the remote file. The requested resource requires user authentication.";
                        case 0x80190194: return "The requested URL does not exist on the
server.";
                        case 0x80190197: return "The user does not have permission to access
the proxy. The proxy requires user authentication.";
                        case 0x8019019E: return "The server cannot process the transfer
request. The Uniform Resource Identifier (URI) in the remote file name is longer than the
server can interpret.";
                        case 0x801901F5: return "The server does not support the functionality
required to fulfill the request. Indicates that BITS uploads are not enabled on the virtual
directory (vdir) on the server.";
                        case 0x801901F7: return "The service is temporarily overloaded and
cannot process the request. Resume the job at a later time.";
                        case 0x801901F8: return "The transfer request timed out while waiting
for a gateway. Resume the job at a later time.";
                        case 0x801901F9: return "The server does not support the HTTP protocol
version specified in the remote file name.";
                        default: return "Unknown error:" + Code;
                    }
                }
            }

            // ### Private methods
            private string Byte2Hex(byte[] Bs)
            {
                //Bs = BitConverter.GetBytes(ErrorCode)
                string RtnStr = "";
                foreach (byte B in Bs)
                {
                    int[] Val = new int[2];
                    Val[0] = B / 16;
                    Val[1] = B % 16;
                    RtnStr = (Val[1] < 10 ? Val[1].ToString() : Val[1] == 10 ? "A" : Val[1] ==
11 ? "B" : Val[1] == 12 ? "C" : Val[1] == 13 ? "D" : Val[1] == 14 ? "E" : "F") + RtnStr;
                    RtnStr = (Val[0] < 10 ? Val[0].ToString() : Val[0] == 10 ? "A" : Val[0] ==
11 ? "B" : Val[0] == 12 ? "C" : Val[0] == 13 ? "D" : Val[0] == 14 ? "E" : "F") + RtnStr;
                }
```

```
                return "0x" + RtnStr;
            }
            private uint ErrorInt2Uint(int ErrorCode)
            {
                // test
                byte[] Bs = BitConverter.GetBytes(ErrorCode);
                uint RtnVal = 0;
                for (int i = 0; i < Bs.Length; i++)
                {
                    RtnVal += (uint)(Bs[i] * Math.Pow(256, i));
                }
                return RtnVal;
            }

            // ### Public methods
            public BitsError(IBackgroundCopyError pError)
            {
                this.Error = pError;
            }
        }

        // ### Private properties
        private class JobCopyCallback : IBackgroundCopyCallback
        {
            public event ProgressDelegate ProgressChanged;
            public event DownloadStateChangedDelegate DownloadStateChanged;
            public event EmptyDelegate DownloadComplete;
            public event ErrorDelegate Error;

            private ulong PrevBytesTransferred = 0;
            private DownloadState PrevState = DownloadState.Undefinded;

            public void JobTransferred(IBackgroundCopyJob pJob)
            {
                pJob.Complete();
                if (DownloadComplete != null)
                    DownloadComplete(null);
            }
            public void JobError(IBackgroundCopyJob pJob, IBackgroundCopyError pError)
            {
                if (Error != null)
                    Error(null, new BitsError(pError));
            }
            public void JobModification(IBackgroundCopyJob pJob, uint dwReserved)
            {
                // Progress
                _BG_JOB_PROGRESS Progress;
                pJob.GetProgress(out Progress);
                if (PrevBytesTransferred != Progress.BytesTransferred && ProgressChanged !=
null)
                    ProgressChanged(null, Progress.BytesTransferred, Progress.BytesTotal);
                PrevBytesTransferred = Progress.BytesTransferred;

                // State
                BG_JOB_STATE State;
                pJob.GetState(out State);
                if ((int)State != (int)PrevState)
                {
                    switch (State)
                    {
                        case BG_JOB_STATE.BG_JOB_STATE_ACKNOWLEDGED: PrevState =
DownloadState.Acknowledged;
                            break;
                        case BG_JOB_STATE.BG_JOB_STATE_CANCELLED: PrevState =
DownloadState.Cancelled;
                            break;
                        case BG_JOB_STATE.BG_JOB_STATE_CONNECTING: PrevState =
DownloadState.Connecting;
                            break;
```

```
                        case BG_JOB_STATE.BG_JOB_STATE_ERROR: PrevState = DownloadState.Error;
                            break;
                        case BG_JOB_STATE.BG_JOB_STATE_QUEUED: PrevState =
DownloadState.Queued;
                            break;
                        case BG_JOB_STATE.BG_JOB_STATE_SUSPENDED: PrevState =
DownloadState.Suspended;
                            break;
                        case BG_JOB_STATE.BG_JOB_STATE_TRANSFERRED: PrevState =
DownloadState.Transferred;
                            break;
                        case BG_JOB_STATE.BG_JOB_STATE_TRANSFERRING: PrevState =
DownloadState.Transferring;
                            break;
                        case BG_JOB_STATE.BG_JOB_STATE_TRANSIENT_ERROR: PrevState =
DownloadState.TransientError;
                            break;
                        default: PrevState = DownloadState.Undefinded;
                            break;
                    }
                    if (DownloadStateChanged != null)
                        DownloadStateChanged(null, PrevState);
                }
            }
        }
        private IBackgroundCopyManager CopyManager;
        private IBackgroundCopyJob CopyJob;
        private Guid JobGuid;
        private JobCopyCallback JobStatus = new JobCopyCallback();

        private DownloadState _State = DownloadState.Undefinded;
        private ulong _BytesTransferred = 0;
        private ulong _BytesTotal = 0;

        // ### Serverclass
        private class Server
        {
            private class WorkerThread
            {
                private class ConnectionThread
                {
                    private Socket Socket;

                    public ConnectionThread(Socket Socket, BITSdownloader BITSdownloader)
                    {
                        this.Socket = Socket;
                        BITSdownloader.DownloadStateChanged += new
DownloadStateChangedDelegate(BITSdownloader_DownloadStateChanged);
                        BITSdownloader.ProgressChanged += new
ProgressDelegate(BITSdownloader_ProgressChanged);
                    }

                    void BITSdownloader_ProgressChanged(BITSdownloader Sender, ulong
BytesTransferred, ulong BytesTotal)
                    {
                        Send("<Progress><Transferred>" + BytesTransferred +
"</Transferred><Total>" + BytesTotal + "</Total></Progress>");
                    }

                    void BITSdownloader_DownloadStateChanged(BITSdownloader Sender,
DownloadState State)
                    {
                        Send("<Download><State>" + State.ToString() + "</State></Download>");
                    }
                    private void Send(string Data)
                    {
                        System.Text.ASCIIEncoding E = new ASCIIEncoding();
                        Socket.Send(E.GetBytes(Data));
                    }
                }
```

```
                private static int _Port;
                private static string _Guid;
                private static Socket _Socket;
                private static BITSdownloader _BITSdownloader;
                private List<ConnectionThread> Connections = new List<ConnectionThread>();

                public WorkerThread(int Port, String Guid, BITSdownloader BITSdownloader)
                {
                    _BITSdownloader = BITSdownloader;
                    _Port = Port;
                    _Guid = Guid;

                    _Socket = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);
                    _Socket.Bind(new IPEndPoint(IPAddress.Any, Port));
                    _Socket.Listen((int)SocketOptionName.MaxConnections);
                }

                public void Start()
                {
                    while (true)
                    {
                        System.Threading.Thread.Sleep(10);
                        Connections.Add(new ConnectionThread(_Socket.Accept(),
_BITSdownloader));
                    }
                }
            }

            private static WorkerThread Worker;
            public static void Start(int Port, String Guid, BITSdownloader BITSdownloader)
            {
                Worker = new WorkerThread(Port, Guid, BITSdownloader);
                System.Threading.Thread T = new System.Threading.Thread(new
System.Threading.ThreadStart(Worker.Start));
                T.Start();

            }
            public static void Send(string Data)
            {

            }
        }

        // ### Public Methods
        public BITSdownloader(string Description)
        {
            CopyManager = (IBackgroundCopyManager)new BackgroundCopyManager();
            CopyManager.CreateJob(Description, BG_JOB_TYPE.BG_JOB_TYPE_DOWNLOAD, out JobGuid,
out CopyJob);
            CopyJob.SetNotifyInterface(JobStatus);
            JobStatus.ProgressChanged += new ProgressDelegate(JobStatus_ProgressChanged);
            JobStatus.DownloadStateChanged += new
DownloadStateChangedDelegate(JobStatus_DownloadStateChanged);
            JobStatus.DownloadComplete += new EmptyDelegate(JobStatus_DownloadComplete);
            JobStatus.Error += new ErrorDelegate(JobStatus_Error);
            CopyJob.SetPriority(BG_JOB_PRIORITY.BG_JOB_PRIORITY_HIGH);
            CopyJob.SetNotifyFlags(BG_JOB_NOTIFICATION_TYPE.BG_NOTIFY_JOB_ERROR |
BG_JOB_NOTIFICATION_TYPE.BG_NOTIFY_JOB_MODIFICATION |
BG_JOB_NOTIFICATION_TYPE.BG_NOTIFY_JOB_TRANSFERRED);
        }

        public BITSdownloader(string Description, int Port, string Guid)
            : this(Description)
        {
            Server.Start(Port, Guid, this);
        }
        public void AddFile(string RemoteURL, string LocalPath)
        {
```

```
            CopyJob.AddFile(RemoteURL, LocalPath);
        }
        public void StartDownload()
        {
            CopyJob.Resume();
        }

        // ### Private events ###
        private void JobStatus_ProgressChanged(BITSdownloader Sender, ulong BytesTransferred,
ulong BytesTotal)
        {
            _BytesTotal = BytesTotal;
            _BytesTransferred = BytesTransferred;
            if (ProgressChanged != null)
                ProgressChanged(this, BytesTransferred, BytesTotal);
        }
        private void JobStatus_DownloadStateChanged(BITSdownloader Sender,
BITSdownloader.DownloadState State)
        {
            _State = State;
            if (DownloadStateChanged != null)
                DownloadStateChanged(this, State);
        }
        private void JobStatus_DownloadComplete(BITSdownloader Sender)
        {
            if (DownloadComplete != null)
                DownloadComplete(this);
        }
        private void JobStatus_Error(BITSdownloader Sender, BitsError Error)
        {
            if (this.Error != null)
                this.Error(this, Error);
        }
    }
}
```

CommonLib

Category.cs

```
using System;
using System.Collections.Generic;
using System.Xml;

namespace AWO.CommonLib
{
    [Serializable]
    public class Category : List<Product>, IXmlObj
    {
        //### Private properties
        private string _GUID = "";
        private string _Name = "";
        private CategoryCollection _ParentCollection;
        //### public properties
        public string GUID
        {
            get { return _GUID; }
            set { _GUID = value; }
        }
        public string Name
        {
            get { return _Name; }
            set { _Name = value; }
        }
        public CategoryCollection ParentCollection
        {
            get { return _ParentCollection; }
        }

        //### Contructors
        public Category(CategoryCollection ParentCollection)
        {
            _ParentCollection = ParentCollection;
        }
        public Category(CategoryCollection ParentCollection, XmlNode
XmlData):this(ParentCollection)
        {
            FromXml(XmlData);
        }

        //### XmlObj Members
        public void FromXml(XmlNode XmlData)
        {
            _Name = XmlData.SelectSingleNode("Name").InnerText;
            _GUID = XmlData.SelectSingleNode("GUID").InnerText;
            this.Clear();
            foreach (XmlNode Node in XmlData.SelectNodes("Products/Product"))
                Add(new Product(this, Node));
        }
        public XmlNode ToXml(XmlDocument XD)
        {
            XmlElement Category = XD.CreateElement("ProductCategory");

            XmlElement Name = XD.CreateElement("Name");
            Name.InnerText = this.Name;
            Category.AppendChild(Name);

            XmlElement GUID = XD.CreateElement("GUID");
            GUID.InnerText = this.GUID;
            Category.AppendChild(GUID);

            XmlElement Products = XD.CreateElement("Products");
            Category.AppendChild(Products);
```

```
            foreach (Product P in this)
                Products.AppendChild(P.ToXml(XD));

            return Category;
        }
        //### Edit, find etc.
        public void RemoveProduct(string GUID)
        {
            Product ProductToRemove = this.Find(delegate(Product P) { return
P.GUID.Equals(GUID); });
            if (ProductToRemove != null)
                this.Remove(ProductToRemove);
        }
    }
}
```

CategoryCollection.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Xml;

namespace AWO.CommonLib
{
    [Serializable]
    public class CategoryCollection : List<Category>, IXmlObj
    {
        //### XML
        public void FromXml(XmlNode XmlData)
        {
            foreach (XmlNode Node in XmlData.SelectNodes("/ProductCategories/ProductCategory"))
                Add(new Category(this, Node));
        }

        public XmlNode ToXml(XmlDocument XD)
        {
            XmlElement CategoryCollection = XD.CreateElement("ProductCategories");

            foreach (Category C in this)
                CategoryCollection.AppendChild(C.ToXml(XD));

            return CategoryCollection;
        }

        //### Edit, find etc.
        public Category FindCategory(string GUID)
        {
            return this.Find(delegate(Category C) { return C.GUID.Equals(GUID); });
        }
        public Product FindProduct(string GUID)
        {
            foreach (Category Cat in this)
                foreach (Product Pro in Cat)
                    if (Pro.GUID.Equals(GUID))
                        return Pro;
            return null;
        }
        public AWO.CommonLib.Version FindVersion(string GUID)
        {
            foreach (Category Cat in this)
                foreach (Product Pro in Cat)
                    foreach (AWO.CommonLib.Version Ver in Pro)
                        if (Ver.GUID.Equals(GUID))
                            return Ver;
            return null;
        }
```

```
        public AWO.CommonLib.Version FindVersion(string ProductGUID, System.Version
VersionNumber)
        {
            foreach (Category Cat in this)
                foreach (Product Pro in Cat)
                    if(Pro.GUID.Equals(ProductGUID))
                        foreach (AWO.CommonLib.Version Ver in Pro)
                            if (Ver.VersionNumber.Equals(VersionNumber))
                                return Ver;
            return null;
        }
    }
}
```

Functions.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Xml;
using System.Text.RegularExpressions;

namespace AWO.CommonLib
{
    public class Functions
    {
        //XML
        public static XmlElement LoadXmlFromData(string Data)
        {
            XmlDocument XD = new XmlDocument();
            XD.LoadXml(Data);
            return (XmlElement)XD.FirstChild;
        }

        //Version
        public static int GetVesionPart(System.Version V, int Index)
        {
            switch (Index)
            {
                case 0:
                    return V.Major;
                case 1:
                    return V.Minor;
                case 2:
                    return V.Build;
                case 3:
                    return V.Revision;
                default:
                    throw new Exception("Index not valid!");
            }
        }
        public static bool IsStringValidVersion(string Str)
        {
            Regex R = new Regex(@"^(\d|([1-9]\d*))(\.\d+){1,3}$", RegexOptions.IgnoreCase);
            return R.IsMatch(Str);
        }
        //File
        public static string GetSizeString(double Size)
        {
            double KB = 1024;
            double MB = Math.Pow(1024, 2);
            double GB = Math.Pow(1024, 3);
            //private static double TB = Math.Pow(1024, 4);
            //private static double PB = Math.Pow(1024, 5);
            //private static double EB = Math.Pow(1024, 6);

            #region Disabled - Support for larger files
            //if (Size >= EB)
```

```
            //{
            //    return Math.Round((Size / EB), 2) + " EB";
            //}
            //else if (Size >= PB)
            //{
            //    return Math.Round((Size / PB), 2) + " PB";
            //}
            //else if (Size >= TB)
            //{
            //    return Math.Round((Size / TB), 2) + " TB";
            //}
            //else
            #endregion

            if (Size >= GB)
                return Math.Round((Size / GB), 2) + " GB";
            else if (Size >= MB)
                return Math.Round((Size / MB), 2) + " MB";
            else if (Size >= KB)
                return Math.Round((Size / KB), 2) + " KB";
            else
                return Size + " B";
        }
        //Sockets
        public static System.Net.Sockets.Socket GetUpdaterSocket(out System.IO.StreamWriter SW,
out System.IO.StreamReader SR)
        {
            System.Net.Sockets.Socket S = new
System.Net.Sockets.Socket(System.Net.Sockets.AddressFamily.InterNetwork,
System.Net.Sockets.SocketType.Stream, System.Net.Sockets.ProtocolType.Tcp);
            S.Connect(new System.Net.IPEndPoint(System.Net.IPAddress.Loopback, 2002));
            if (S.Connected)
            {
                System.Net.Sockets.NetworkStream NS = new System.Net.Sockets.NetworkStream(S);
                SW = new System.IO.StreamWriter(NS);
                SR = new System.IO.StreamReader(NS);
                return S;
            }
            throw new Exception("Could not connect to updatersocket");
        }
        public static string GetDataFromUpdater(string Command)
        {
            System.IO.StreamReader SR;
            System.IO.StreamWriter SW;
            System.Net.Sockets.Socket S = Functions.GetUpdaterSocket(out SW, out SR);
            SW.WriteLine(Command);
            SW.Flush();
            //string RtnData = SR.ReadLine();
            string RtnData = SR.ReadLine().Replace("¤", "\n");
            S.Close();
            return RtnData;
        }
    }
}
```

Product.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Xml;

namespace AWO.CommonLib
{
    [Serializable]
    public class Product : List<Version>, IXmlObj
    {
        //### Private properties
```

```
        private string _GUID = "";
        private string _Name = "";
        private string _Description = "";
        private Category _ParentCategory;
        //### Public Properties
        public string GUID
        {
            get { return _GUID; }
            set { _GUID = value; }
        }
        public string Name
        {
            get { return _Name; }
            set { _Name = value; }
        }
        public string Description
        {
            get { return _Description; }
            set { _Description = value; }
        }
        public Category ParentCategory
        {
            get { return _ParentCategory; }
        }
        public Version CurrentVersion
        {
            get
            {
                //get active the version with the highest version number
                List<Version> ActiveVersions = this.FindAll(delegate(Version PV) { return
PV.Active; });
                if (ActiveVersions == null || ActiveVersions.Count == 0)
                    return null;

                Version CurrentPV = ActiveVersions[0];
                for (int i = 1; i < ActiveVersions.Count; i++)
                {
                    if (ActiveVersions[i].CompareTo(CurrentPV) == 1)
                        CurrentPV = ActiveVersions[i];
                }
                return CurrentPV;
            }
        }
        //### Contructors
        public Product(Category ParentCategory)
        {
            _ParentCategory = ParentCategory;
        }
        public Product(Category ParentCategory, XmlNode XmlData) : this(ParentCategory)
        {
            FromXml(XmlData);
        }

        //### XmlObj Members
        public void FromXml(XmlNode XmlData)
        {
            _Name = XmlData.SelectSingleNode("Name").InnerText;
            _GUID = XmlData.SelectSingleNode("GUID").InnerText;
            _Description = XmlData.SelectSingleNode("Description").InnerText;
            this.Clear();
            foreach (XmlNode Node in XmlData.SelectNodes("Versions/Version"))
                Add(new Version(this, Node));
        }
        public XmlNode ToXml(XmlDocument XD)
        {
            XmlElement Product = XD.CreateElement("Product");

            XmlElement Name = XD.CreateElement("Name");
            Name.InnerText = this.Name;
            Product.AppendChild(Name);
```

```
            XmlElement GUID = XD.CreateElement("GUID");
            GUID.InnerText = this.GUID;
            Product.AppendChild(GUID);

            XmlElement Description = XD.CreateElement("Description");
            Description.InnerText = this.Description;
            Product.AppendChild(Description);

            XmlElement Versions = XD.CreateElement("Versions");
            Product.AppendChild(Versions);
            foreach(Version V in this)
                Versions.AppendChild(V.ToXml(XD));

            return Product;
        }

        //### Edit, find etc.
        public List<Version> GetVersionsNewerThan(System.Version VersionNumber)
        {
            return FindAll(delegate(Version V) { return
V.VersionNumber.CompareTo(VersionNumber) == 1; });
        }
        public Version FindVersion(string GUID)
        {
            return this.Find(delegate(Version V) { return V.GUID.Equals(GUID); });
        }
        public Version FindVersion(System.Version VersionNumber)
        {
            return this.Find(delegate(Version V) { return
V.VersionNumber.Equals(VersionNumber); });
        }
        public void RemoveVersion(string GUID)
        {
            AWO.CommonLib.Version VersionToRemove = this.Find(delegate(AWO.CommonLib.Version V)
{ return V.GUID.Equals(GUID); });
            if (VersionToRemove != null)
                this.Remove(VersionToRemove);
        }

        //### ToString Etc.
        public override string ToString()
        {
            return Name;
        }
    }
}
```

Version.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Xml;

namespace AWO.CommonLib
{
    [Serializable]
    public class Version : IXmlObj, IComparable
    {
        //### Private properties
        private Product _Product;
        private List<VersionReference> _RequiredVersions = new List<VersionReference>();
        //### public properties
        public Product Product
        {
            get { return _Product; }
            set { _Product = value; }
```

```
        }
        public List<VersionReference> RequiredVersions
        {
            get { return _RequiredVersions; }
        }
        //### Private Xml properties
        private string _GUID = "";
        private bool _Active = true;
        private System.Version _VersionNumber = new System.Version();
        private string _Description = "";
        private string _FilePath = "";
        private uint _FileSize = 0;
        private string _CommandArgument = "";
        //### XmlObj Properties (public)
        public string GUID
        {
            get { return _GUID; }
            set { _GUID = value; }
        }
        public System.Version VersionNumber
        {
            get { return _VersionNumber; }
            set { _VersionNumber = value; }
        }
        public bool Active
        {
            get { return _Active; }
            set { _Active = value; }
        }
        public string Description
        {
            get { return _Description; }
            set { _Description = value; }
        }
        public string FilePath
        {
            get { return _FilePath; }
            set { _FilePath = value; }
        }
        public uint FileSize
        {
            get { return _FileSize; }
            set { _FileSize = value; }
        }
        public string CommandArgument
        {
            get { return _CommandArgument; }
            set { _CommandArgument = value; }
        }
        //### Contructors
        public Version(Product Product)
        {
            this.Product = Product;
        }
        public Version(Product Product, XmlNode XmlData) : this(Product)
        {
            FromXml(XmlData);
        }

        //### XmlObj Members
        public void FromXml(XmlNode XmlData)
        {
            _VersionNumber = new
System.Version(XmlData.SelectSingleNode("VersionNumber").InnerText);
            _Active = XmlData.SelectSingleNode("Active").InnerText.Equals(true.ToString()) ?
true : false;
            _GUID = XmlData.SelectSingleNode("GUID").InnerText;
            _Description = XmlData.SelectSingleNode("Description").InnerText;
            _FilePath = XmlData.SelectSingleNode("FilePath").InnerText;
            _FileSize = uint.Parse(XmlData.SelectSingleNode("FileSize").InnerText);
```

```
            _CommandArgument = XmlData.SelectSingleNode("CommandArgument").InnerText;
            RequiredVersions.Clear();
            foreach (XmlNode Node in XmlData.SelectNodes("RequiredVersions/VersionReference"))
                RequiredVersions.Add(new VersionReference(Node));
        }
        public XmlNode ToXml(XmlDocument XD)
        {
            XmlElement Version = XD.CreateElement("Version");

            XmlElement VersionNumber = XD.CreateElement("VersionNumber");
            VersionNumber.InnerText = this.VersionNumber.ToString();
            Version.AppendChild(VersionNumber);

            XmlElement Active = XD.CreateElement("Active");
            Active.InnerText = this.Active.ToString();
            Version.AppendChild(Active);

            XmlElement GUID = XD.CreateElement("GUID");
            GUID.InnerText = this.GUID;
            Version.AppendChild(GUID);

            XmlElement Description = XD.CreateElement("Description");
            Description.InnerText = this.Description;
            Version.AppendChild(Description);

            XmlElement FilePath = XD.CreateElement("FilePath");
            FilePath.InnerText = this.FilePath;
            Version.AppendChild(FilePath);

            XmlElement FileSize = XD.CreateElement("FileSize");
            FileSize.InnerText = this.FileSize.ToString();
            Version.AppendChild(FileSize);

            XmlElement CommandArgument = XD.CreateElement("CommandArgument");
            CommandArgument.InnerText = this.CommandArgument;
            Version.AppendChild(CommandArgument);

            XmlElement RequiredVersions = XD.CreateElement("RequiredVersions");
            Version.AppendChild(RequiredVersions);
            foreach (VersionReference VR in this.RequiredVersions)
                RequiredVersions.AppendChild(VR.ToXml(XD));

            return Version;
        }

        // CompareTo, ToString Etc.
        public override string ToString()
        {
            return this.VersionNumber.ToString();
        }

        public int CompareTo(object Obj)
        {
            if (Obj is AWO.CommonLib.Version)
                return CompareTo(((AWO.CommonLib.Version)Obj).VersionNumber);
            else if (Obj is System.Version)
                return CompareTo((System.Version)Obj);
            else if (Obj is string)
                return CompareTo((string)Obj);
            else
                throw new Exception("Unhandled type: '" + Obj.GetType().ToString() + "'");
        }

        public int CompareTo(System.Version OtherVersion)
        {
            for (int i = 0; i < 3; i++)
            {
                int This = ((Functions.GetVesionPart(this.VersionNumber, i) <= 0) ? 0 :
Functions.GetVesionPart(this.VersionNumber, i));
```

```
                int Other = ((Functions.GetVesionPart(OtherVersion, i) <= 0) ? 0 :
Functions.GetVesionPart(OtherVersion, i));

                if (This > Other)
                    return 1;
                else if (This < Other)
                    return -1;
            }
            return 0;
        }
        public int CompareTo(string OtherVersionString)
        {
            return CompareTo(new System.Version(OtherVersionString));
        }
    }
}
```

VersionReference.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Xml;

namespace AWO.CommonLib
{
    [Serializable]
    public class VersionReference : IXmlObj
    {
        //### Private properties
        private System.Version _VersionNumber = new System.Version();
        private string _ProductGUID = Guid.Empty.ToString();
        //### Public properties
        public System.Version VersionNumber
        {
            get { return _VersionNumber; }
            set { _VersionNumber = value; }
        }
        public string ProductGUID
        {
            get { return _ProductGUID; }
            set { _ProductGUID = value; }
        }

        //### Contructors
        public VersionReference(System.Xml.XmlNode Element)
        {
            FromXml(Element);
        }
        public VersionReference(string ProductGUID, System.Version VersionNumber)
        {
            this.ProductGUID = ProductGUID;
            this.VersionNumber = VersionNumber;
        }
        //### XmlObj Members
        public void FromXml(System.Xml.XmlNode XmlData)
        {
            _VersionNumber = new
System.Version(XmlData.SelectSingleNode("VersionNumber").InnerText);
            _ProductGUID = XmlData.SelectSingleNode("ProductGUID").InnerText;
        }

        public System.Xml.XmlNode ToXml(System.Xml.XmlDocument XD)
        {
            XmlElement VersionReference = XD.CreateElement("VersionReference");

            XmlElement VersionNumber = XD.CreateElement("VersionNumber");
            VersionNumber.InnerText = this.VersionNumber.ToString();
```

```
            VersionReference.AppendChild(VersionNumber);

            XmlElement ProductGUID = XD.CreateElement("ProductGUID");
            ProductGUID.InnerText = this.ProductGUID;
            VersionReference.AppendChild(ProductGUID);

            return VersionReference;
        }
    }
}
```

XmlObj.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Xml;

namespace AWO.CommonLib
{
    public interface IXmlObj
    {
        void FromXml(XmlNode Element);
        XmlNode ToXml(XmlDocument XD);
    }
}
```

UpdaterCommonLib

Job.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Xml;

namespace AWO.UpdaterCommonLib
{
    public class Job : AWO.CommonLib.IXmlObj
    {
        public class Result
        {
            public Nullable<bool> Success = null;
            public string ErrorMsg = "";

            public Result()
            {

            }
        }
        private string _GUID;
        private string _Name;
        private Version _Version;
        private string _File;
        private string _Size;
        private string _Argument;

        public Result InstallResult = new Result();

        public string GUID
        {
            get { return _GUID; }
        }
        public string Name
        {
            get { return _Name; }
        }
        public Version Version
        {
            get { return _Version; }
        }
        public string File
        {
            get { return _File; }
        }
        public string Size
        {
            get { return _Size; }
        }
        public string Argument
        {
            get { return _Argument; }
        }
        public Job(XmlElement XmlData)
        {
            FromXml(XmlData);
        }
        public Job(string GUID, string Name, string Version, string File, string Size, string
Argument)
        {
            this._GUID = GUID;
            this._Name = Name;
            this._Version = new Version(Version);
            this._File = File;
```

```
            this._Size = Size;
            this._Argument = Argument;
        }

        public void FromXml(XmlNode Element)
        {
            _GUID = Element.SelectSingleNode("/Job/GUID").InnerText;
            _Name = Element.SelectSingleNode("/Job/Name").InnerText;
            _Version = new System.Version(Element.SelectSingleNode("/Job/Version").InnerText);
            _File = Element.SelectSingleNode("/Job/File").InnerText;
            _Size = Element.SelectSingleNode("/Job/Size").InnerText;
            _Argument = Element.SelectSingleNode("/Job/Argument").InnerText;
        }

        public XmlNode ToXml(XmlDocument XD)
        {
            XmlElement Job = XD.CreateElement("Job");

            XmlElement GUID = XD.CreateElement("GUID");
            GUID.InnerText = this.GUID;
            Job.AppendChild(GUID);

            XmlElement Name = XD.CreateElement("Name");
            Name.InnerText = this.Name;
            Job.AppendChild(Name);

            XmlElement Version = XD.CreateElement("Version");
            Version.InnerText = this.Version.ToString();
            Job.AppendChild(Version);

            XmlElement File = XD.CreateElement("File");
            File.InnerText = this.File;
            Job.AppendChild(File);

            XmlElement Size = XD.CreateElement("Size");
            Size.InnerText = this.Size;
            Job.AppendChild(Size);

            XmlElement Argument = XD.CreateElement("Argument");
            Argument.InnerText = this.Argument;
            Job.AppendChild(Argument);

            return Job;
        }
    }
}
```

Settings.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Xml;

namespace AWO.UpdaterCommonLib
{
    public class Settings : AWO.CommonLib.IXmlObj
    {
        public class Product
        {
            public string GUID;
            public string ProductName;

            public Product(string GUID, string ProductName)
            {
                this.ProductName = ProductName;
                this.GUID = GUID;
            }
```

```
        }
        public enum UpdateDays : int
        {
            EveryDay = 0,
            Monday = 1,
            Tuesday = 2,
            Wednesday = 3,
            Thursday = 4,
            Friday = 5,
            Saturday = 6,
            Sunday = 7
        }
        public enum ServiceStates : int
        {
            AWODisabled = 0,
            AutoUpdateServiceDisabled = 1,
            AutoUpdateServiceEnabled = 2
        }

        public string CompanyName;
        public string CustomerGuid;
        public ServiceStates ServiceState;
        public UpdateDays UpdateDay;
        public int UpdateHour;

        public List<Product> DisabledProducts = new List<Product>();

        public Settings()
        {
        }
        public Settings(XmlDocument XD)
        {
            FromXml(XD);
        }

        public void FromXml(XmlNode Element)
        {
            CompanyName = Element.SelectSingleNode("/Settings/CompanyName").InnerText;
            CustomerGuid = Element.SelectSingleNode("/Settings/CustomerGuid").InnerText;
            ServiceState =
(ServiceStates)int.Parse(Element.SelectSingleNode("/Settings/ServiceState").InnerText);
            UpdateDay =
(UpdateDays)int.Parse(Element.SelectSingleNode("/Settings/UpdateDay").InnerText);
            UpdateHour = int.Parse(Element.SelectSingleNode("/Settings/UpdateHour").InnerText);

            foreach (XmlNode Product in
Element.SelectNodes("/Settings/DisabledProducts/Product"))
            {
                string ProductName = Product.SelectSingleNode("ProductName").InnerText;
                string ProductGUID = Product.SelectSingleNode("ProductGUID").InnerText;
                DisabledProducts.Add(new Product(ProductGUID, ProductName));
            }
        }

        public XmlNode ToXml(XmlDocument XD)
        {
            XmlElement Root = XD.CreateElement("Settings");
            XD.AppendChild(Root);

            XmlElement CompanyName = XD.CreateElement("CompanyName");
            CompanyName.InnerText = this.CompanyName;
            Root.AppendChild(CompanyName);

            XmlElement CustomerGuid = XD.CreateElement("CustomerGuid");
            CustomerGuid.InnerText = this.CustomerGuid;
            Root.AppendChild(CustomerGuid);

            XmlElement ServiceState = XD.CreateElement("ServiceState");
            ServiceState.InnerText = ((int)this.ServiceState).ToString();
            Root.AppendChild(ServiceState);
```

```
            XmlElement UpdateDay = XD.CreateElement("UpdateDay");
            UpdateDay.InnerText = ((int)this.UpdateDay).ToString();
            Root.AppendChild(UpdateDay);

            XmlElement UpdateHour = XD.CreateElement("UpdateHour");
            UpdateHour.InnerText = ((int)this.UpdateHour).ToString();
            Root.AppendChild(UpdateHour);

            // DisabledProducts
            XmlElement DisabledProducts = XD.CreateElement("DisabledProducts");
            Root.AppendChild(DisabledProducts);
            foreach (Product P in this.DisabledProducts)
            {
                XmlElement Product = XD.CreateElement("Product");
                DisabledProducts.AppendChild(Product);

                XmlElement ProductName = XD.CreateElement("ProductName");
                ProductName.InnerText = P.ProductName;
                Product.AppendChild(ProductName);

                XmlElement ProductGUID = XD.CreateElement("ProductGUID");
                ProductGUID.InnerText = P.GUID;
                Product.AppendChild(ProductGUID);
            }

            return XD;
        }
    }
}
```

Update.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Xml;

namespace AWO.UpdaterCommonLib
{
    public class UpdateRequirement : AWO.CommonLib.IXmlObj
    {
        public string Name;
        public string GUID;
        public System.Version VersionNumber;

        public UpdateRequirement(string Name, string GUID, System.Version VersionNumber)
        {
            this.Name = Name;
            this.GUID = GUID;
            this.VersionNumber = VersionNumber;
        }
        public UpdateRequirement(XmlNode Node)
        {
            FromXml(Node);
        }

        public void FromXml(XmlNode XmlData)
        {
            Name = XmlData.SelectSingleNode("Name").InnerText;
            GUID = XmlData.SelectSingleNode("GUID").InnerText;
            VersionNumber = new Version(XmlData.SelectSingleNode("VersionNumber").InnerText);
        }
        public XmlNode ToXml(XmlDocument XD)
        {
            XmlElement UpdateRequirement = XD.CreateElement("UpdateRequirement");

            XmlElement Name = XD.CreateElement("Name");
```

```
            Name.InnerText = this.Name;
            UpdateRequirement.AppendChild(Name);

            XmlElement GUID = XD.CreateElement("GUID");
            GUID.InnerText = this.GUID;
            UpdateRequirement.AppendChild(GUID);

            XmlElement VersionNumber = XD.CreateElement("VersionNumber");
            VersionNumber.InnerText = this.VersionNumber.ToString();
            UpdateRequirement.AppendChild(VersionNumber);

            return UpdateRequirement;
        }
    }
    public class Update : AWO.CommonLib.IXmlObj
    {
        public string Name;
        public string GUID;
        public string Description;
        public string CategoryName;
        public string CategoryGUID;
        public System.Version VersionNumber;
        public List<UpdateRequirement> Requirements = new List<UpdateRequirement>();

        private Update(string Name, string GUID, string Description, System.Version
VersionNumber, string CategoryName, string CategoryGUID)
        {
            this.Name = Name;
            this.GUID = GUID;
            this.Description = Description;
            this.VersionNumber = VersionNumber;
            this.CategoryName = CategoryName;
            this.CategoryGUID = CategoryGUID;
        }
        public Update(AWO.CommonLib.Version Ver, AWO.CommonLib.CategoryCollection Categories) :
this(Ver.Product.Name,Ver.Product.GUID,Ver.Product.Description,Ver.VersionNumber,Ver.Product.Pa
rentCategory.Name,Ver.Product.ParentCategory.GUID)
        {
            foreach (AWO.CommonLib.VersionReference VR in Ver.RequiredVersions)
                Requirements.Add(new
UpdateRequirement(Categories.FindProduct(VR.ProductGUID).Name, VR.ProductGUID,
VR.VersionNumber));
        }
        public Update(XmlNode Node)
        {
            FromXml(Node);
        }
        public void FromXml(XmlNode XmlData)
        {
            Name = XmlData.SelectSingleNode("Name").InnerText;
            GUID = XmlData.SelectSingleNode("GUID").InnerText;
            Description = XmlData.SelectSingleNode("Description").InnerText;
            VersionNumber = new Version(XmlData.SelectSingleNode("VersionNumber").InnerText);
            CategoryName = XmlData.SelectSingleNode("CategoryName").InnerText;
            CategoryGUID = XmlData.SelectSingleNode("CategoryGUID").InnerText;
            Requirements.Clear();
            foreach (XmlNode Node in XmlData.SelectNodes("Requirements/UpdateRequirement"))
                Requirements.Add(new UpdateRequirement(Node));
        }
        public XmlNode ToXml(XmlDocument XD)
        {
            XmlElement Update = XD.CreateElement("Update");

            XmlElement Name = XD.CreateElement("Name");
            Name.InnerText = this.Name;
            Update.AppendChild(Name);

            XmlElement GUID = XD.CreateElement("GUID");
            GUID.InnerText = this.GUID;
            Update.AppendChild(GUID);
```

```
            XmlElement Description = XD.CreateElement("Description");
            Description.InnerText = this.Description;
            Update.AppendChild(Description);

            XmlElement VersionNumber = XD.CreateElement("VersionNumber");
            VersionNumber.InnerText = this.VersionNumber.ToString();
            Update.AppendChild(VersionNumber);

            XmlElement CategoryName = XD.CreateElement("CategoryName");
            CategoryName.InnerText = this.CategoryName;
            Update.AppendChild(CategoryName);

            XmlElement CategoryGUID = XD.CreateElement("CategoryGUID");
            CategoryGUID.InnerText = this.CategoryGUID;
            Update.AppendChild(CategoryGUID);

            XmlElement Requirements = XD.CreateElement("Requirements");
            Update.AppendChild(Requirements);
            foreach (UpdateRequirement UP in this.Requirements)
                Requirements.AppendChild(UP.ToXml(XD));

            return Update;
        }
    }
}
```

UpdaterConfig

Program.cs

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace UpdateConfig
{
    static class Program
    {
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new ConfigForm());
        }
    }
}
```

ProductSettings.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Xml;
using System.Windows.Forms;
using AWO.UpdaterCommonLib;

namespace UpdateConfig
{
    public partial class ProductSettings : Form
    {
        ConfigForm CF;
        List<CheckBox> ProductCheckBoxes = new List<CheckBox>();
        public ProductSettings(ConfigForm CF)
        {
            this.CF = CF;
            InitializeComponent();
            int YPos = 26;

            XmlDocument InstalledProductsXML = new XmlDocument();

InstalledProductsXML.LoadXml(AWO.CommonLib.Functions.GetDataFromUpdater("GetInstalledProducts:"
));
            List<AWO.CommonLib.VersionReference> InstalledProducts = new
List<AWO.CommonLib.VersionReference>();
            foreach (XmlNode N in
InstalledProductsXML.SelectNodes("/Products/VersionReference"))
                InstalledProducts.Add(new AWO.CommonLib.VersionReference(N));

            foreach (AWO.CommonLib.VersionReference VR in InstalledProducts)
            {
                CheckBox CB = new CheckBox();
                //Get product from VS
                string ProductString = AWO.CommonLib.Functions.GetDataFromUpdater("GetProduct:"
+ VR.ProductGUID);
                if (ProductString != "")
                {
                    XmlDocument TempXD = new XmlDocument();
                    TempXD.LoadXml(ProductString);
```

```
                    AWO.CommonLib.Product P = new AWO.CommonLib.Product(null,
TempXD.SelectSingleNode("Product"));
                    CB.Text = P.Name;
                }
                else
                {
                    CB.Text = "Unknown product";
                    MessageBox.Show("ukendt produkt installeret\n" + VR.ProductGUID);
                }
                CB.Left = 15;
                CB.Width = 235;
                CB.Top = YPos;
                CB.Name = VR.ProductGUID.ToString();
                YPos += 26;
                if (CF.CurrentSettings.DisabledProducts.Find(delegate(Settings.Product P) {
return P.GUID.Equals(VR.ProductGUID.ToString()); }) != null)
                    CB.Checked = true;
                this.Controls.Add(CB);
                ProductCheckBoxes.Add(CB);
            }
            this.Width = 250;
            this.Height = YPos + 60;
        }

        private void Ok_Click(object sender, EventArgs e)
        {
            CF.CurrentSettings.DisabledProducts.Clear();
            foreach (CheckBox CB in ProductCheckBoxes)
                if (CB.Checked)
                    CF.CurrentSettings.DisabledProducts.Add(new Settings.Product(CB.Name,
CB.Text));
            this.Close();
        }
    }
}
```

ConfigForm.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Xml;
using System.Windows.Forms;
using AWO.UpdaterCommonLib;

namespace UpdateConfig
{
    public partial class ConfigForm : Form
    {
        private class DropDownItem<T>
        {
            public string Text;
            public T Value;

            public DropDownItem(string Text, T Value)
            {
                this.Text = Text;
                this.Value = Value;
            }

            public override string ToString()
            {
                return Text;
            }
        }
```

```
        public Settings CurrentSettings;
        private ProductSettings PS;

        public ConfigForm()
        {
            // Get Settings
            try
            {
                XmlDocument XD = new XmlDocument();
                XD.LoadXml(AWO.CommonLib.Functions.GetDataFromUpdater("GetSettings:"));
                CurrentSettings = new Settings(XD);
            }
            catch (Exception E)
            {
                MessageBox.Show("Kunne ikke hente AWO indstillingerne, tjek at 'Webtop updater'
servicen kører", "AWO", MessageBoxButtons.OK, MessageBoxIcon.Error);
                Application.ExitThread();
                return;
            }

            PS = new ProductSettings(this);
            InitializeComponent();

            // Initialize dropdowns
            UpdateDays.Items.Add(new DropDownItem<Settings.UpdateDays>("Hver dag",
Settings.UpdateDays.EveryDay));
            UpdateDays.Items.Add(new DropDownItem<Settings.UpdateDays>("Hver mandag",
Settings.UpdateDays.Monday));
            UpdateDays.Items.Add(new DropDownItem<Settings.UpdateDays>("Hver tirsdag",
Settings.UpdateDays.Tuesday));
            UpdateDays.Items.Add(new DropDownItem<Settings.UpdateDays>("Hver onsdag",
Settings.UpdateDays.Wednesday));
            UpdateDays.Items.Add(new DropDownItem<Settings.UpdateDays>("Hver torsdag",
Settings.UpdateDays.Thursday));
            UpdateDays.Items.Add(new DropDownItem<Settings.UpdateDays>("Hver fredag",
Settings.UpdateDays.Friday));
            UpdateDays.Items.Add(new DropDownItem<Settings.UpdateDays>("Hver lørdag",
Settings.UpdateDays.Saturday));
            UpdateDays.Items.Add(new DropDownItem<Settings.UpdateDays>("Hver søndag",
Settings.UpdateDays.Sunday));

            for (int i = 0; i < 24; i++)
                UpdateTime.Items.Add(new DropDownItem<int>(i.ToString() + ":00", i));

            // set values
            // Companyname
            CompanyName.Text = CurrentSettings.CompanyName;
            // Service state
            switch (CurrentSettings.ServiceState)
            {
                case Settings.ServiceStates.AWODisabled:
                    RadioAWODeactivated.Checked = true;
                    break;
                case Settings.ServiceStates.AutoUpdateServiceDisabled:
                    RadioUpdateServiceDeactivated.Checked = true;
                    break;
                case Settings.ServiceStates.AutoUpdateServiceEnabled:
                    RadioUpdateServiceEnabled.Checked = true;
                    break;
            }
            // Updateday
            foreach (DropDownItem<Settings.UpdateDays> Item in UpdateDays.Items)
                if (Item.Value.Equals(CurrentSettings.UpdateDay))
                    UpdateDays.SelectedItem = Item;
            // Updatehour
            foreach (DropDownItem<int> Item in UpdateTime.Items)
                if (Item.Value.Equals(CurrentSettings.UpdateHour))
                    UpdateTime.SelectedItem = Item;
        }
```

```
        private void Radio_CheckedChanged(object sender, EventArgs e)
        {
            if (RadioUpdateServiceEnabled.Checked)
            {
                UpdateDays.Enabled = true;
                UpdateTime.Enabled = true;
                AdminProducts.Enabled = true;
            }
            else
            {
                UpdateDays.Enabled = false;
                UpdateTime.Enabled = false;
                AdminProducts.Enabled = false;
            }
        }
        private void Cancel_Click(object sender, EventArgs e)
        {
            this.Close();
        }
        private void Ok_Click(object sender, EventArgs e)
        {
            // validate

            // Set'n'Save settings
            CurrentSettings.CompanyName = CompanyName.Text;

            if (RadioAWODeactivated.Checked)
                CurrentSettings.ServiceState = Settings.ServiceStates.AWODisabled;
            else if (RadioUpdateServiceDeactivated.Checked)
                CurrentSettings.ServiceState =
Settings.ServiceStates.AutoUpdateServiceDisabled;
            else if (RadioUpdateServiceEnabled.Checked)
                CurrentSettings.ServiceState = Settings.ServiceStates.AutoUpdateServiceEnabled;

            CurrentSettings.UpdateDay =
((DropDownItem<Settings.UpdateDays>)UpdateDays.SelectedItem).Value;

            CurrentSettings.UpdateHour = ((DropDownItem<int>)UpdateTime.SelectedItem).Value;

            AWO.CommonLib.Functions.GetDataFromUpdater("SaveSettings:" +
CurrentSettings.ToXml(new XmlDocument()).OuterXml);
            this.Close();
        }
        private void AdminProducts_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
        {
            PS.ShowDialog();
        }
    }
}
```

Updater

Updater.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Net;
using System.Net.Sockets;
using System.Xml;
using AWO.CommonLib;
using AWO.UpdaterCommonLib;
using Microsoft.Win32;

namespace AWO.Updater
{
    public enum LogLevel : int
    {
        Debug = 0,
        Info = 1,
        Warning = 2,
        Error = 3
    }
    public delegate void LogDelegate(string Message, LogLevel DebugLevel);
    public class Updater
    {
        private LogDelegate _Log;
        public void Log(string Message, LogLevel DebugLevel)
        {
            if (_Log != null)
                _Log(Message, DebugLevel);
        }
        public Updater(LogDelegate LogCallBack)
        {
            _Log = LogCallBack;
            Log("Updater initialized", LogLevel.Info);
        }

        public InstallFSM FSM;
        public WebServer WS;
        public VersionsServer.VersionsService VS = new
AWO.Updater.VersionsServer.VersionsService();
        private System.Timers.Timer ScheduleTimer = new System.Timers.Timer(1000 * 60); //
Every Min

        public void Start()
        {
            // VS
            VS.Credentials = System.Net.CredentialCache.DefaultCredentials;

            // Start State machine and FSM
            FSM = new InstallFSM(this);
            FSM.InstallStarted += new InstallFSM.InstallLogDelegate(FSM_InstallStarted);
            FSM.InstallComplete += new InstallFSM.InstallLogDelegate(FSM_InstallComplete);

            WS = new WebServer(82, Guid.NewGuid().ToString());
            WS.ParsePage = new WebServer.ParsePageDelegate(FSM.ParsePage);
            WS.PageRequest += new WebServer.PageRequestDelegate(WS_PageRequest);
            WS.Start();

            // Start ScheduleTimer
            ScheduleTimer.Elapsed += new
System.Timers.ElapsedEventHandler(ScheduleTimer_Elapsed);
            ScheduleTimer.Start();

            // Start Server for Interprocess communication... using sockets, in own thread
```

```
            System.Threading.Thread T = new System.Threading.Thread(new
System.Threading.ThreadStart(ClientListnerThread));
            T.Start();
        }

        private void FSM_InstallStarted(List<Job> Jobs)
        {
            Settings S = SettingsLib.GetSettings();
            XmlDocument InstalledVersionReferences = new XmlDocument();
            XmlDocument UpdateVersionReferences = new XmlDocument();

            XmlElement VersionReferences;

            VersionReferences = InstalledVersionReferences.CreateElement("VersionReferences");
            InstalledVersionReferences.AppendChild(VersionReferences);
            foreach (VersionReference VR in GetInstalledProducts())
                VersionReferences.AppendChild(VR.ToXml(InstalledVersionReferences));

            VersionReferences = UpdateVersionReferences.CreateElement("VersionReferences");
            UpdateVersionReferences.AppendChild(VersionReferences);
            foreach (Job Job in Jobs)
                VersionReferences.AppendChild((new VersionReference(Job.GUID,
Job.Version)).ToXml(UpdateVersionReferences));

            VS.LogBeforeUpdate(S.CustomerGuid, WS.InstanceGUID,
InstalledVersionReferences.OuterXml, UpdateVersionReferences.OuterXml);
        }

        private void FSM_InstallComplete(List<Job> Jobs)
        {
            Settings S = SettingsLib.GetSettings();
            XmlDocument InstalledVersionReferences = new XmlDocument();
            XmlElement VersionReferences;

            VersionReferences = InstalledVersionReferences.CreateElement("VersionReferences");
            InstalledVersionReferences.AppendChild(VersionReferences);
            foreach (VersionReference VR in GetInstalledProducts())
                VersionReferences.AppendChild(VR.ToXml(InstalledVersionReferences));

            VS.LogAfterUpdate(S.CustomerGuid, WS.InstanceGUID,
InstalledVersionReferences.OuterXml);
        }

        private void ClientListnerThread()
        {
            Socket ServerSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);
            ServerSocket.Bind(new IPEndPoint(IPAddress.Loopback, 2002));
            while (true)
            {
                ServerSocket.Listen(1);
                Socket S = ServerSocket.Accept();
                Log("Client Connected", LogLevel.Debug);
                //Console.WriteLine("Client connected");
                ClientHandler CH = new ClientHandler(S, this);
                CH.Disconnected += new ClientHandler.DisconnectedDelegate(CH_Disconnected);
                System.Threading.Thread T = new System.Threading.Thread(new
System.Threading.ThreadStart(CH.Start));
                T.Start();
            }
        }

        private void ScheduleTimer_Elapsed(object sender, System.Timers.ElapsedEventArgs e)
        {
            Settings S = SettingsLib.GetSettings();
            if (S.UpdateDay == Settings.UpdateDays.EveryDay ||
                (int)DateTime.Now.DayOfWeek == (int)S.UpdateDay)
            {
                // right day... check for right time
                if (DateTime.Now.Hour == S.UpdateHour && DateTime.Now.Minute == 0)
```

```
                {
                    // Get updates
                    List<Update> Updates = new List<Update>();
                    XmlDocument XD = new XmlDocument();
                    XD.LoadXml(Functions.GetDataFromUpdater("GetUpdates"));
                    foreach (XmlNode Node in XD.SelectNodes("/Updates/Update"))
                        Updates.Add(new Update(Node));

                    // Build jobslist
                    XmlDocument JobXD = new XmlDocument();
                    XmlElement Jobs = JobXD.CreateElement("Jobs");
                    JobXD.AppendChild(Jobs);
                    foreach (Update U in Updates)
                        if(U.Requirements.Count==0)
                            Jobs.AppendChild(new VersionReference(U.GUID,
U.VersionNumber).ToXml(JobXD));

                    string ProcessGUID = Functions.GetDataFromUpdater("NewJob:" +
JobXD.OuterXml);
                    Log("Running autoupdate\nProcessGUID:" + ProcessGUID, LogLevel.Info);
                }
            }
        }
        private void WS_PageRequest(Uri Url)
        {
            if (Url.LocalPath.Equals("/Status.htm") && FSM.CurrentState ==
InstallFSM.FSMStates.NotStarted)
                FSM.CurrentState = InstallFSM.FSMStates.Download;
        }
        private void CH_Disconnected(ClientHandler Client)
        {
            Log("Client Disconnected", LogLevel.Debug);
        }

        public List<AWO.CommonLib.VersionReference> GetInstalledProducts()
        {
            List<AWO.CommonLib.VersionReference> InstalledProducts = new
List<AWO.CommonLib.VersionReference>();

            // Get all known products from versionserver
            string[] AllProductGUIDS = VS.GetAllProductGUIDs();

            // Scan for installed products
            RegistryKey RegKey = Registry.LocalMachine;
            RegKey = RegKey.OpenSubKey(@"SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall");

            foreach (string ProductGUID in AllProductGUIDS)
            {
                RegistryKey CurKey = RegKey.OpenSubKey("{" + ProductGUID.ToUpper() + "}");
                if (CurKey != null)
                    InstalledProducts.Add(new AWO.CommonLib.VersionReference(ProductGUID, new
System.Version(CurKey.GetValue("DisplayVersion").ToString())));
            }

            return InstalledProducts;
        }
    }
}
```

SettingsLib.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Xml;
using AWO.UpdaterCommonLib;

namespace AWO.Updater
```

```
{
    class SettingsLib
    {
        public static void SaveSettings(string XmlData)
        {
            XmlDocument XD = new XmlDocument();
            XD.LoadXml(XmlData);
            XD.Save("AWOSettings.xml");
        }
        public static string GetSettingsAsXml()
        {
            return GetSettings().ToXml(new XmlDocument()).OuterXml;
        }
        public static Settings GetSettings()
        {
            Settings S;
            if (System.IO.File.Exists("AWOSettings.xml"))
            {
                XmlDocument XD = new XmlDocument();
                XD.Load("AWOSettings.xml");
                S = new Settings(XD);
            }
            else
            {
                // Default settings
                S = new Settings();
                S.CompanyName = "";
                S.CustomerGuid = Guid.NewGuid().ToString();
                S.ServiceState = Settings.ServiceStates.AWODisabled;
                S.UpdateDay = Settings.UpdateDays.EveryDay;
                S.UpdateHour = 0;
            }
            return S;
        }
    }
}
```

ClientHandler.cs

```
using System;
using System.Xml;
using System.Collections.Generic;
using System.Diagnostics;
using System.Text;
using System.IO;
using System.Net;
using System.Net.Sockets;
using AWO.UpdaterCommonLib;

namespace AWO.Updater
{
    class ClientHandler
    {
        public delegate void DisconnectedDelegate(ClientHandler Client);
        public event DisconnectedDelegate Disconnected;

        private Socket S;
        private Updater Updater;
        public ClientHandler(Socket S, Updater Updater)
        {
            this.S = S;
            this.Updater = Updater;
        }
        public void Start()
        {
            NetworkStream NS = new NetworkStream(S, System.IO.FileAccess.ReadWrite, true);
            StreamReader SR = new StreamReader(NS);
            StreamWriter SW = new StreamWriter(NS);
```

```
            try
            {
                while (S.Connected && !SR.EndOfStream)
                {
                    SW.WriteLine(HandleCommand(SR.ReadLine()).Replace("\n","¤"));
                    SW.Flush();
                }

                if (S.Connected)
                    NS.Close();
                if (Disconnected != null)
                    Disconnected(this);
            }
            catch (Exception E)
            {
                Updater.Log(E.ToString(),LogLevel.Warning);
                //Console.WriteLine(E.InnerException);
            }
        }

        private string HandleCommand(string Data)
        {
            Updater.Log("Command:" + Data, LogLevel.Debug);
            //Console.WriteLine("Command:" + Data);
            if (Data.StartsWith("NewJob"))
                return HandleCommand_NewJob(Data.Substring(Data.IndexOf(":") + 1));
            else if (Data.StartsWith("GetSettings"))
                return HandleCommand_GetSettings();
            else if (Data.StartsWith("SaveSettings"))
                return HandleCommand_SaveSettings(Data.Substring(Data.IndexOf(":") + 1));
            else if (Data.StartsWith("GetUpdates"))
                return HandleCommand_GetUpdates();
            else if (Data.StartsWith("GetInstalledProducts"))
                return HandleCommand_GetInstalledProducts();
            else if (Data.StartsWith("GetProduct"))
                return HandleCommand_GetProduct(Data.Substring(Data.IndexOf(":") + 1));
            else
                return "Unknown command";
        }

        // Update Job stuff
        private string HandleCommand_NewJob(string Data)
        {
            if (!(Updater.FSM.CurrentState == InstallFSM.FSMStates.NotStarted ||
Updater.FSM.CurrentState == InstallFSM.FSMStates.Finish))
            {
                Updater.Log("Trying to start a new update job, while an existing job is already
running", LogLevel.Warning);
                //Updater.EventLog.WriteEntry("Trying to start a new update job, while an
existing job is already running", EventLogEntryType.Warning, 3);
                //Console.WriteLine("A job is already running, using existing process");
            }
            else
            {
                // Extract VersionReferences from data
                List<string> JobsVR = new List<string>();
                XmlDocument JobsXML = new XmlDocument();
                JobsXML.InnerXml = Data;
                foreach (XmlElement XMLVR in JobsXML.SelectNodes("/Jobs/VersionReference"))
                    JobsVR.Add((new AWO.CommonLib.VersionReference(XMLVR)).ToXml(new
XmlDocument()).OuterXml);

                // Build joblist
                List<Job> Jobs = new List<Job>();
                foreach (string JobStr in Updater.VS.GetJobs(JobsVR.ToArray()))
                {
                    Jobs.Add(new Job(AWO.CommonLib.Functions.LoadXmlFromData(JobStr)));
                }

                if (Jobs.Count > 0)
```

```
                {
                    Updater.WS.InstanceGUID = Guid.NewGuid().ToString();
                    Updater.FSM.SetNewJobBatch(Jobs);
                    Updater.FSM.CurrentState = InstallFSM.FSMStates.Download;

                }
            }
            return Updater.WS.InstanceGUID;
        }
        private string HandleCommand_SaveSettings(string Data)
        {
            SettingsLib.SaveSettings(Data);
            Settings S = SettingsLib.GetSettings();
            Updater.VS.UpdateCompanyName(S.CustomerGuid, S.CompanyName);
            return "";
        }
        private string HandleCommand_GetSettings()
        {
            return SettingsLib.GetSettingsAsXml();
        }
        private string HandleCommand_GetUpdates()
        {
            List<string> InstalledProducts = new List<string>();
            foreach (AWO.CommonLib.VersionReference VR in Updater.GetInstalledProducts())
                InstalledProducts.Add(VR.ToXml(new XmlDocument()).OuterXml);

            // Get updates from versionserver
            string[] UpdatesData = Updater.VS.GetUpdates(InstalledProducts.ToArray());
            List<Update> Updates = new List<Update>();
            foreach (string S in UpdatesData)
                Updates.Add(new Update(AWO.CommonLib.Functions.LoadXmlFromData(S)));

            XmlDocument XD = new XmlDocument();
            XmlElement XE = XD.CreateElement("Updates");
            XD.AppendChild(XE);

            foreach (Update U in Updates)
                XE.AppendChild(U.ToXml(XD));

            return XD.OuterXml;
        }
        private string HandleCommand_GetInstalledProducts()
        {
            XmlDocument XD = new XmlDocument();
            XmlElement XE = XD.CreateElement("Products");
            XD.AppendChild(XE);

            foreach (AWO.CommonLib.VersionReference VR in Updater.GetInstalledProducts())
                XE.AppendChild(VR.ToXml(XD));
            return XD.OuterXml;
        }
        private string HandleCommand_GetProduct(string Data)
        {
            return Updater.VS.GetProduct(Data);
        }
    }
}
```

Webserver.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Threading;
using System.IO;
using System.Net;
using System.Net.Sockets;
using System.Collections.Specialized;
```

```
namespace AWO.Updater
{
    public class WebServer
    {
        // ### Public Events
        public delegate void PageRequestDelegate(Uri Url);
        public event PageRequestDelegate PageRequest;

        // ### Private properties
        private const string SERVERID = "Webtop ServerInstaller Webserver";
        private Thread ServerThread;
        private Socket ServerSocket;

        private int Port;

        // ### Public properties
        public string InstanceGUID;

        // ### Public Delegates
        public delegate string ParsePageDelegate(string Page);
        public ParsePageDelegate ParsePage;

        // ### Public functions
        public void Start()
        {
            if (ServerThread != null)
                ServerThread.Abort();

            ServerThread = new Thread(new ThreadStart(Loop));
            ServerThread.IsBackground = true;
            ServerThread.Start();
        }
        public void Stop()
        {
            ServerSocket.Close();
            ServerThread.Abort();
            ServerThread = null;
        }
        public WebServer(int Port, string InstanceGUID)
        {
            this.Port = Port;
            this.InstanceGUID = InstanceGUID;
        }

        // ### Private functions
        private void Loop()
        {
            ServerSocket = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);
            ServerSocket.Bind(new IPEndPoint(IPAddress.Any, Port));

            while (true)
            {
                // Accept connection
                ServerSocket.Listen(1);
                Socket S = ServerSocket.Accept();
                NetworkStream NS = new NetworkStream(S, System.IO.FileAccess.ReadWrite, true);
                StreamReader SR = new StreamReader(NS);

                // Get Command
                string Command = GetCommand(SR);
                Uri Url = ExtraceUriFromCommand(Command, S.LocalEndPoint);

                // Only process command if InstanceGUID is correct
                if (InstanceGUID.Equals(ExtraceQueryStringValue(Url.Query, "GUID")))
                {
                    // Raise event
                    if (PageRequest != null)
                        PageRequest(Url);
```

```
                    // Handle command
                    if (Url.LocalPath.Replace("/", "\\").Substring(Url.LocalPath.Replace("/",
"\\").LastIndexOf('.') + 1).Equals("gif"))
                    { // Gif image
                        SendOk(NS, false, false);
                        Stream ImgStream =
System.Reflection.Assembly.GetExecutingAssembly().GetManifestResourceStream("AWO.Updater.Webser
ver.HTML." + Url.LocalPath.Replace("/", ""));
                        byte[] Buff = new byte[ImgStream.Length];
                        ImgStream.Read(Buff, 0, (int)ImgStream.Length);
                        SendBytes(Buff, NS);
                    }
                    else
                    {
                        SendOk(NS, true, true);
                        string Page = new
StreamReader(System.Reflection.Assembly.GetExecutingAssembly().GetManifestResourceStream("AWO.U
pdater.Webserver.HTML." + Url.LocalPath.Replace("/", ""))).ReadToEnd();

                        // Parse page, if parser is defined
                        if (ParsePage != null)
                            Page = ParsePage(Page);

                        Page = Page.Replace("{INSTANCEGUID}", InstanceGUID);

                        SendString(Page, NS);
                    }
                }
                else
                {
                    SendString("<h1>ERROR</h1>Instance GUID did not match!!!", NS);
                }

                // Close connection
                NS.Flush();
                NS.Close();
            }
        }
        private void SendBytes(byte[] Data, NetworkStream NS)
        {
            try
            {
                NS.Write(Data, 0, Data.Length);
            }
            catch (Exception e)
            {
                //Console.WriteLine(e.ToString());
            }
        }
        private void SendString(String Msg, NetworkStream NS)
        {
            byte[] Buf;
            Buf = System.Text.Encoding.ASCII.GetBytes(Msg);

            try
            {
                NS.Write(Buf, 0, Buf.Length);
            }
            catch (Exception e)
            {
                Console.WriteLine(e.ToString());
            }
        }
        private void SendOk(NetworkStream NS, bool DisableCache, bool Text)
        {
            SendString("HTTP/1.1 200 OK\r\n", NS);
            SendString(string.Format("Date:{0}\r\n", DateTime.Now), NS);
            SendString(string.Format("Server:{0}\r\n", SERVERID), NS);
            if (Text)
```

```
                SendString("Content-Type: text/html; charset=utf-8\r\n", NS);
            if (DisableCache)
            {
                SendString("Expires: Sat, 1 Jan 2005 00:00:00 GMT\r\n", NS);
                SendString("Cache-Control: no-cache, must-revalidate\r\n", NS);
                SendString("Pragma: no-cache\r\n", NS);
            }

            // End header
            SendString("\r\n", NS);
        }
        private string GetCommand(StreamReader SR)
        {
            string Buf;
            string Command = null;
            StringDictionary Parameters = new StringDictionary(); // Not used for anything...
might be usefull later
            try
            {
                while ((Buf = SR.ReadLine()) != null && Buf.Length > 0)
                {
                    if (Command == null)
                        Command = Buf;
                    else // Paramters
                    {
                        int colonIdx = Buf.IndexOf(":");
                        if (colonIdx > 0)
                            Parameters.Add(Buf.Substring(0, colonIdx).Trim(),
Buf.Substring(colonIdx + 1).Trim());
                    }
                }
            }
            catch (Exception E)
            {
                Console.WriteLine(E.ToString());
            }
            return Command;
        }
        private Uri ExtraceUriFromCommand(string Command, EndPoint LocalEndPoint)
        {
            // Extract file part of command
            Command = Command.Trim();
            Command = Command.Substring(Command.IndexOf(" ") + 1);
            Command = Command.Substring(0, Command.IndexOf(" "));

            return new Uri("http://" + LocalEndPoint.ToString() + Command);
        }
        private string LoadFile(string Path)
        {
            if (System.IO.File.Exists(Path))
                return System.IO.File.ReadAllText(Path);
            else
                return "<H1>Could not find file:" + Path + "</h1>";
        }

        private string ExtraceQueryStringValue(string QueryString, string Key)
        {
            // Temp solution...
            QueryString = QueryString.Replace("?", "&");
            foreach (string S in QueryString.Split('&'))
                if (S.Split('=')[0].Equals(Key))
                    return S.Split('=')[1];
            return null;
        }
    }
}
```

Status.htm

```
<?xml version="1.0" encoding="iso-8859-1"?>
<html>
<head>
    <title>Webtop Updater - Codename: AWO</title>
    <BASE target="_self">
    <style type="text/css">
        Body
        {
            background-color:rgb(240, 240, 240);
        }

        Div, TD
        {
            font-family:Verdana;
            font-size:12px;
        }

        TD.RightCell
        {
            float:left;
            background-color:rgb(180, 180, 180);
            border-right:solid 2px #000000;
            font-size:14px;
            font-weight:bold;
            height:expression(document.body.clientHeight);
            width:200px;
            padding:20px 0px 0px 10px;
        }
        TD.MainArea
        {
            height:expression(document.body.clientHeight);
            width:expression(document.body.clientWidth - 210);
        }
        Div.OverviewArea
        {
            display:none;
            height:300px;
            margin:30px 0px 0px 30px;
            width:expression(document.body.clientWidth - 220);
        }
        Div.ButtonArea
        {
            display:none;
            height:20px;
            margin:30px 30px 0px 30px;
            width:expression(document.body.clientWidth - 220);
        }
        Div.LogArea
        {
            height:expression(250);
            width:expression(document.body.clientWidth - 270);
            background-color:rgb(180, 180, 180);
            margin:30px 30px 0px 30px;
            border:solid 1px #000000;
        }

        Div.DownloadArea
        {
            margin:30px 30px 0px 30px;
            height:expression(50);
            width:expression(document.body.clientWidth - 270);
        }
    </style>
</head>
<body style="margin:0px;">
    <table cellpadding="0" cellspacing="0">
        <tr>
            <td class="RightCell" valign="top">
```

```
                <span id="State_1" style="display:none;">&middot; Starter<br /></span>
                <span id="State_2">&middot; Downloader<br /></span>
                <span id="State_3">&middot; Installer<br /></span>
                <span id="State_4">&middot; Afslutter<br /></span>
                <span id="State_5">&middot; Oversigt<br /></span>
            </td>
            <td class="MainArea" valign="top">
                <div id="OverviewArea" class="OverviewArea">

<b style="font-size:larger;font-weight:bold;">Installations oversigt</b><br />
<br />
<b><img style="vertical-align:middle;" src="successlg.gif?GUID={INSTANCEGUID}" /> Installeret
opdateringer (<span id="InstalledUpdateCount"></span>)</b><br />
<span id="InstalledUpdates"></span>
<br />
<b><img style="vertical-align:middle;" src="failedlg.gif?GUID={INSTANCEGUID}" /> Mislykket
opdateringer (<span id="FailedUpdateCount"></span>)</b><br />
<span id="FailedUpdates"></span>
                </div>
                <div id="ButtonArea" class="ButtonArea">
                    <a href="#" onclick="window.close();">Luk</a>
                </div>
                <div id="LogArea" class="LogArea">
                </div>

                <div id="DownloadArea" class="DownloadArea">
                    <div id="StatusText">Downloading</div>

                    <div style="position:absolute;width:100%;Border:solid 1px
#000000;background-color:white;"></div>
                    <div id="BarDiv" style="position:absolute;width:0%;Border:solid 1px
#000000;background-color:red;">
                        <div id="SubBarDiv" style="width:0%;background-color:white;"></div>
                    </div>
                    <div id="TextDiv" style="position:absolute;width:100%;text-
align:center;">40%</div>
                </div>
            </td>
        </tr>
    </table>

    <script>
        var PrevLogText;
        var CurrentLogEntry = 0;
        var StatusAnimation = 0;
        var ProgressAnimation = 0;
        var ProgressAnimationRunning = false;
        var Done = false;
        function SetStatusText(Text)
        {
            document.getElementById("StatusText").innerText = Text;
        }
        function SetState(Index)
        {
            for(i=1;i<=4;i++)
                document.getElementById("State_" + i).style.color = "";

            if(document.getElementById("State_" + Index) == null)
                alert(Index);
            document.getElementById("State_" + Index).style.color = "Red";
        }

        function SetProgress(Percent, HidePercent, Animation)
        {
            // Abort if trying to start animation again
            if(Animation && ProgressAnimationRunning)
                return;

            // Start animation
            if(Animation)
```

```
                ProgressAnimationRunning = true;

            // Stop animation if a Percent is given
            if(Percent != null)
                ProgressAnimationRunning = false;

            if(Percent == null && ProgressAnimationRunning)
            {
                ProgressAnimation = (ProgressAnimation+1)%201;
                Percent = ProgressAnimation;
                window.setTimeout("SetProgress(null,true);",20);
            }
            else if(Percent == null && !ProgressAnimationRunning) // Abort
            {
                return;
            }

            if(HidePercent == null)
                HidePercent = false;

            if(!HidePercent)
                document.getElementById("TextDiv").innerText = Percent + "%";
            else
                document.getElementById("TextDiv").innerText = "";

            if(Percent<=100)
            {
                document.getElementById("SubBarDiv").style.width = 0 + "%";
                document.getElementById("SubBarDiv").style.borderRight = "";
                document.title = Percent;
                document.getElementById("BarDiv").style.width = Percent + "%";
            }
            else if(Percent>100 && Percent<=200)
            {
                document.getElementById("SubBarDiv").style.width = (Percent-100) + "%";
                document.getElementById("SubBarDiv").style.borderRight = "solid 1px #000000";
            }
        }
        function AddToLog(Text, Status)
        {
            if(Text != PrevLogText)
            {
                if(CurrentLogEntry>0)
                    document.getElementById("LogEntryStatus_" + CurrentLogEntry).innerText =
"Done";
                PrevLogText = Text;
                CurrentLogEntry++;
                var NewLogElement = '<table style="width:100%;"><tr><td>' + Text + '</td><td
id="LogEntryStatus_' + CurrentLogEntry + '" align=right>' + Status + '</td></tr></table>';
                document.getElementById("LogArea").innerHTML += NewLogElement;
            }
            else
            {
                document.getElementById("LogEntryStatus_" + CurrentLogEntry).innerText =
Status;
            }
        }
        function HandleResponse(Response)
        {
            var State = Response.split('|')[0];
            var Progress = Response.split('|')[1];

            switch(State)
            {
                case "1" :
                    SetStatusText("Collect Information");
                    SetProgress(Progress);
                    AddToLog("Collecting updateinformation", "Scanning...");
                    break;
                case "2" :
```

```
                    var JobNumber = Response.split('|')[2];
                    var NumberOfJobs = Response.split('|')[3];
                    var Size = Response.split('|')[4];
                    var PacketName = Response.split('|')[5];
                    var Status = Response.split('|')[6];
                    AddToLog("(" + JobNumber + ":" + NumberOfJobs + ") Downloading " +
PacketName + " (" + Size + ")", Status + "...");
                    SetProgress(Progress);
                    SetStatusText("Downloading...");
                    break;
                case "3":
                    var JobNumber = Response.split('|')[2];
                    var NumberOfJobs = Response.split('|')[3];
                    var PacketName = Response.split('|')[4];
                    AddToLog("(" + JobNumber + ":" + NumberOfJobs + ") Installing " +
PacketName, "Installing...");
                    SetProgress(null, true, true);
                    SetStatusText("Installing...");
                    break;
                case "4":
                    Done = true;
                    document.getElementById("LogArea").style.display="none";
                    document.getElementById("DownloadArea").style.display="none";
                    document.getElementById("OverviewArea").style.display="block";
                    document.getElementById("ButtonArea").style.display="block";

                    var InstalledUpdateCount = 0;
                    var FailedUpdateCount = 0;
                    var InstalledUpdates = "";
                    var FailedUpdates = "";

                    for(i=2;i<Response.split('|').length-1;i++)
                    {
                        if(Response.split('|')[i].indexOf("1") == 0)
                        {
                            InstalledUpdateCount++;
                            InstalledUpdates += Response.split('|')[i].substr(1) + "<br>";
                        }
                        else
                        {
                            FailedUpdateCount++;
                            FailedUpdates += Response.split('|')[i].substr(1) + "<br>";
                        }
                    }

                    document.getElementById("InstalledUpdateCount").innerHTML =
InstalledUpdateCount;
                    document.getElementById("InstalledUpdates").innerHTML = InstalledUpdates;
                    document.getElementById("FailedUpdateCount").innerHTML = FailedUpdateCount;
                    document.getElementById("FailedUpdates").innerHTML = FailedUpdates;

                    break;
                case "5":
                    if(!Done)
                    {
                        alert(Response.split('|')[1]);
                        Done = true;
                    }
                    break;
                default :
                    break;
            }

            SetState(State);
        }
        function Run()
        {
            var requester = null;

            if (requester != null && requester.readyState != 0 && requester.readyState != 4)
```

```
                requester.abort();

            try{
                requester = new XMLHttpRequest();
            }
            catch (error){
                try{
                    requester = new ActiveXObject("Microsoft.XMLHTTP");
                }
                catch (error){
                    requester = null;
                    return false;
                }
            }

            requester.open("GET", "Status.ajax?GUID={INSTANCEGUID}",true);

            requester.onreadystatechange = function()
            {
                if (requester.readyState == 4 && requester.status == 200)
                    HandleResponse(requester.responseText);
            }

            requester.send(null);
            if(!Done)
                window.setTimeout("Run()",500);
        }
        Run();
    </script>
</body>
</html>
```

InstallFSM.cs

```
using System;
using System.Diagnostics;
using System.Collections.Generic;
using System.Text;
using AWO.UpdaterCommonLib;

namespace AWO.Updater
{
    public class InstallFSM
    {
        // ### Public events and delegates
        public delegate void InstallLogDelegate(List<Job> Jobs);
        public event InstallLogDelegate InstallComplete;
        public event InstallLogDelegate InstallStarted;

        // ### Public properties
        public InstallFSM(Updater Updater)
        {
            this.Updater = Updater;
        }
        public void SetNewJobBatch(List<Job> Jobs)
        {
            this.Jobs = Jobs;
            CurrentState = FSMStates.NotStarted;
        }
        public enum FSMStates : int
        {
            NotStarted = 0,
            CollectInfo = 1,
            Download = 2,
            Install = 3,
            Finish = 4,
            Error = 5
        }
        public FSMStates CurrentState
```

```
        {
            get { return _CurrentState; }
            set
            {
                _CurrentState = value;
                System.Threading.Thread T;
                switch (value)
                {
                    case FSMStates.CollectInfo:
                        _Progress = 0;
                        T = new System.Threading.Thread(new
System.Threading.ThreadStart(CollectInfoThread));
                        T.Start();
                        break;
                    case FSMStates.Download:
                        _Progress = 0;
                        _CurrentJob = 0;
                        T = new System.Threading.Thread(new
System.Threading.ThreadStart(DownloadThread));
                        T.Start();
                        break;
                    case FSMStates.Install:
                        _Progress = 0;
                        T = new System.Threading.Thread(new
System.Threading.ThreadStart(InstallThread));
                        T.Start();
                        break;
                    case FSMStates.Finish:
                        _Progress = 0;
                        T = new System.Threading.Thread(new
System.Threading.ThreadStart(FinishThread));
                        T.Start();
                        break;
                    default:
                        break;
                }
            }
        }
        public int Progress
        {
            get
            {
                //Console.WriteLine("Getting progress:" + _Progress.ToString());
                return _Progress;
            }
        }

        // ### Private properties
        private Updater Updater;
        private List<Job> Jobs;
        private FSMStates _CurrentState;
        private int _Progress = 0;
        private int _CurrentJob;
        private BITS.BITSdownloader.DownloadState _CurrentDownloadState =
BITS.BITSdownloader.DownloadState.Undefinded;
        private string DownloadDir
        {
            get
            {
                if (!System.IO.Directory.Exists(System.Windows.Forms.Application.StartupPath +
"\\DownloadCache"))

System.IO.Directory.CreateDirectory(System.Windows.Forms.Application.StartupPath +
"\\DownloadCache");
                return System.Windows.Forms.Application.StartupPath + "\\DownloadCache";
            }
        }
        private string Error = "";

        // ### PageRequest parsing
```

```
        private string StatusString
        {
            get
            {
                string RtnStr = ((int)CurrentState).ToString() + "|";
                RtnStr += Progress.ToString() + "|";

                switch (CurrentState)
                {
                    case FSMStates.CollectInfo:
                        break;
                    case FSMStates.Download:
                        RtnStr += _CurrentJob + 1 + "|"; //2
                        RtnStr += Jobs.Count + "|"; //3
                        //RtnStr +=
AWO.CommonLib.Functions.GetSizeString((double)(Jobs[_CurrentJob].Size)) + "|"; //4
                        RtnStr += Jobs[_CurrentJob].Size + "|"; //4
                        RtnStr += Jobs[_CurrentJob].Name + "|"; //5
                        RtnStr += _CurrentDownloadState ==
BITS.BITSdownloader.DownloadState.Undefinded ? "Starting" : _CurrentDownloadState.ToString() +
"|"; //6
                        break;
                    case FSMStates.Install:
                        RtnStr += _CurrentJob + 1 + "|"; //2
                        RtnStr += Jobs.Count + "|"; //3
                        RtnStr += Jobs[_CurrentJob].Name + "|"; //4
                        break;
                    case FSMStates.Finish:
                        foreach (Job J in Jobs)
                            if (J.InstallResult.Success.HasValue)
                                RtnStr += (J.InstallResult.Success == true ? "1" : "0") +
J.Name + " (" + J.Version.ToString() + ")|";
                        break;
                    case FSMStates.Error:
                        RtnStr += Error;
                        break;
                    default:
                        break;
                }

                return RtnStr;
            }
        }
        public string ParsePage(string Page)
        {
            Page = Page.Replace("[TIME]", DateTime.Now.ToLongTimeString());
            Page = Page.Replace("[STATUSSTRING]", StatusString);
            return Page;
        }

        // ### "State-Threads"
        private void CollectInfoThread()
        {
            Random R = new Random();
            for (int i = 0; i <= 100; i += 5)
            {
                _Progress = i;

                System.Threading.Thread.Sleep(R.Next(100, 800));
            }
            CurrentState = FSMStates.Download;
        }
        private void DownloadThread()
        {
            if (InstallStarted != null)
                InstallStarted(Jobs);
            for (int i = 0; i < Jobs.Count; i++)
            {
                _CurrentJob = i;
                Job CurrentJob = Jobs[i];
```

```
                BITS.BITSdownloader Downloader = new BITS.BITSdownloader(CurrentJob.Name);

                // Create temp dir for download
                if (!System.IO.Directory.Exists(DownloadDir + "\\" + CurrentJob.GUID))
                    System.IO.Directory.CreateDirectory(DownloadDir + "\\" + CurrentJob.GUID);

                string FileName = CurrentJob.File.Substring(CurrentJob.File.LastIndexOf("/") +
1);
                string LocalFile = DownloadDir + "\\" + CurrentJob.GUID + "\\" + FileName;
                string RemoteFile = CurrentJob.File;

                Updater.Log("Downloading file:" + RemoteFile + " to " + LocalFile,
LogLevel.Info);
                //Console.WriteLine("New job:" + LocalFile);
                //Console.WriteLine("Remote file:" + RemoteFile);

                Downloader.AddFile(RemoteFile, LocalFile);

                Downloader.DownloadStateChanged += new
BITS.BITSdownloader.DownloadStateChangedDelegate(Downloader_DownloadStateChanged);
                Downloader.ProgressChanged += new
BITS.BITSdownloader.ProgressDelegate(Downloader_ProgressChanged);
                Downloader.Error += new BITS.BITSdownloader.ErrorDelegate(Downloader_Error);

                Downloader.StartDownload();
                while ((int)Downloader.Progress < 100)
                {
                    System.Threading.Thread.Sleep(100);
                }
            }
            if(CurrentState != FSMStates.Error)
                CurrentState = FSMStates.Install;
        }
        private void InstallThread()
        {
            Random Rnd = new Random();
            for (int i = 0; i < Jobs.Count; i++)
            {
                _Progress = 0;
                _CurrentJob = i;
                Job CurrentJob = Jobs[i];

                // Locate setup file
                string File = DownloadDir + "\\" + CurrentJob.GUID + "\\" +
CurrentJob.File.Substring(CurrentJob.File.LastIndexOf("/") + 1);
                _Progress = 5;

                // Wait for install file to be ready (need to be Acknowledged by BITS);
                System.Threading.Thread.Sleep(100);
                while (!System.IO.File.Exists(File))
                    System.Threading.Thread.Sleep(100);

                // Install
                System.Diagnostics.Process P = new System.Diagnostics.Process();
                P.StartInfo.FileName = "\"" + File + "\"";
                //P.StartInfo.Arguments = "/s";
                P.StartInfo.Arguments = CurrentJob.Argument;
                P.Start();
                P.WaitForExit();

                if (P.ExitCode == 0)
                {
                    // All good
                    CurrentJob.InstallResult.Success = true;
                    Updater.Log("Product successfully updated\r\n" + CurrentJob.Name + "(" +
CurrentJob.Version + ")", LogLevel.Info);
                    //Service.EventLog.WriteEntry("Product successfully updated\r\n" +
CurrentJob.Name + "(" + CurrentJob.Version + ")", EventLogEntryType.Information, 4);
                }
                else
```

```
                {
                    // Something went wrong!!!
                    CurrentJob.InstallResult.Success = false;
                    CurrentJob.InstallResult.ErrorMsg = "Updateringen af " + CurrentJob.Name +
" fejlede, prøv at køre opdateringen igen";
                    Updater.Log("Failed to update\r\n" + CurrentJob.Name + "(" +
CurrentJob.Version + ")",LogLevel.Info);
                    //Service.EventLog.WriteEntry("Failed to update\r\n" + CurrentJob.Name +
"(" + CurrentJob.Version + ")", EventLogEntryType.Warning, 5);
                }

                _Progress = 90;

                // Check Log

                _Progress = 100;
            }
            CurrentState = FSMStates.Finish;
        }
        private void FinishThread()
        {
            _Progress = 0;
            // Delete temp files
            for (int i = 0; i < Jobs.Count; i++)
            {
                _CurrentJob = i;
                Job CurrentJob = Jobs[i];
                if (System.IO.Directory.Exists(DownloadDir + "\\" + CurrentJob.GUID))
                    System.IO.Directory.Delete(DownloadDir + "\\" + CurrentJob.GUID, true);
                _Progress = ((i + 1) * 100) / Jobs.Count;
                System.Threading.Thread.Sleep(1250);
            }
            _Progress = 100;
            if (InstallComplete != null)
                InstallComplete(Jobs);
        }

        // ### Internal events handlers
        private void Downloader_Error(BITS.BITSdownloader Sender, BITS.BITSdownloader.BitsError
Err)
        {
            Updater.Log("Download error:" + Err.Code + "\n" + Err.Description,LogLevel.Error);
            Error = "Download error:" + Err.Code + "\n" + Err.Description;
            CurrentState = FSMStates.Error;
            //Console.WriteLine("Error:" + Err.Code);
            //Console.WriteLine(Err.Description);
        }
        private void Downloader_ProgressChanged(BITS.BITSdownloader Sender, ulong
BytesTransferred, ulong BytesTotal)
        {
            Updater.Log("Downloadprogress:" + BytesTransferred + "/" + BytesTotal + "(" +
(BytesTransferred * 100) / BytesTotal + "%)", LogLevel.Debug);
            //Console.WriteLine("Progress:" + BytesTransferred + "/" + BytesTotal + "(" +
(BytesTransferred * 100) / BytesTotal + "%)");
            _Progress = (int)Sender.Progress;
        }
        private void Downloader_DownloadStateChanged(BITS.BITSdownloader Sender,
BITS.BITSdownloader.DownloadState State)
        {
            _CurrentDownloadState = State;
            Updater.Log("DownloadState Changed:" + State.ToString(), LogLevel.Debug);
            //Console.WriteLine("DownloadState Changed:" + State.ToString());
        }
    }
}
```

UpdaterConsole

Program.cs

```
using System;
using System.Collections.Generic;
using System.Text;

namespace AWO.UpdaterConsole
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Starter Updater...");
            AWO.Updater.Updater U = new AWO.Updater.Updater(Log);
            U.Start();
            Console.ReadLine();
        }

        static void Log(string Message, AWO.Updater.LogLevel DebugLevel)
        {
            Console.WriteLine(Message);
        }
    }
}
```

UpdaterService

Service.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.ServiceProcess;
using System.Text;
using System.IO;

namespace Updater
{
    public partial class Service : ServiceBase
    {
        public Service()
        {
            InitializeComponent();
            if (!System.Diagnostics.EventLog.SourceExists("Webtop AWO"))
                System.Diagnostics.EventLog.CreateEventSource("Webtop AWO", "Webtop");
            EventLog.Source = "Webtop AWO";
            EventLog.Log = "Webtop";
        }
        protected override void OnStart(string[] args)
        {
            AWO.Updater.Updater U = new AWO.Updater.Updater(Log);
            U.Start();
            Log("Webservice startede", AWO.Updater.LogLevel.Info);
        }
        protected void Log(string Message, AWO.Updater.LogLevel DebugLevel)
        {
            if (DebugLevel == AWO.Updater.LogLevel.Debug)
                return;
            EventLogEntryType T = EventLogEntryType.Information;
            switch(DebugLevel)
            {
                case AWO.Updater.LogLevel.Info :
                    T = EventLogEntryType.Information;
                    break;
                case AWO.Updater.LogLevel.Warning:
                    T = EventLogEntryType.Warning;
                    break;
                case AWO.Updater.LogLevel.Error:
                    T = EventLogEntryType.Error;
                    break;
            }
            EventLog.WriteEntry(EventLog.Source, Message, T);
            Console.WriteLine(Message);
        }
    }
}
```

ProjectInstaller.Designer.cs

```
namespace Updater
{
    partial class ProjectInstaller
    {
        private System.ComponentModel.IContainer components = null;

        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
```

```
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        private void InitializeComponent()
        {
            this.serviceProcessInstaller1 = new
System.ServiceProcess.ServiceProcessInstaller();
            this.serviceInstaller = new System.ServiceProcess.ServiceInstaller();
            if (!System.IO.File.Exists("serviceinfo.txt"))
                System.IO.File.WriteAllText("serviceinfo.txt","webtopone\\bsc|******");

            this.serviceProcessInstaller1.Account = System.ServiceProcess.ServiceAccount.User;
            this.serviceProcessInstaller1.Username =
System.IO.File.ReadAllText("serviceinfo.txt").Split('|')[0];
            this.serviceProcessInstaller1.Password =
System.IO.File.ReadAllText("serviceinfo.txt").Split('|')[1];
            this.serviceInstaller.ServiceName = "Webtop Updater service";
            this.serviceInstaller.StartType = System.ServiceProcess.ServiceStartMode.Automatic;
            this.serviceInstaller.AfterInstall += new
System.Configuration.Install.InstallEventHandler(this.serviceInstaller1_AfterInstall);
            this.Installers.AddRange(new System.Configuration.Install.Installer[] {
            this.serviceProcessInstaller1,
            this.serviceInstaller});

        }

        #endregion

        private System.ServiceProcess.ServiceProcessInstaller serviceProcessInstaller1;
        private System.ServiceProcess.ServiceInstaller serviceInstaller;
    }
}
```

KundeServer (Website)

UpdaterControl.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Collections.Generic;
using System.Xml;
using AWO.CommonLib;
using AWO.UpdaterCommonLib;

namespace AWO.Kunde
{
    public class UpdaterControl : WebControl
    {
        // ### Nested Classes
        protected class ContentCell : HtmlTableCell
        {
            // ### ContentCell Constructors
            public ContentCell()
                : base()
            {
                base.Style[HtmlTextWriterStyle.BorderWidth] = "0";
                base.Style[HtmlTextWriterStyle.TextAlign] = "left";
            }

            public ContentCell(string InnerHTML)
                : this()
            {
                this.InnerHtml = InnerHTML;
            }
        }
        protected class HeaderCell : ContentCell
        {
            // ### HeaderCell Constructors
            public HeaderCell()
                : base()
            {
                this.Style[HtmlTextWriterStyle.FontSize] = "16px";
                this.Style[HtmlTextWriterStyle.FontWeight] = "bold";
            }

            public HeaderCell(string InnerHTML)
                : this()
            {
                this.InnerHtml = InnerHTML;
            }
        }
        protected class UpdateCell : ContentCell
        {
            static int Id; //Used for Show/Hide JavaScript, PrePend ClientId if multiple
instances must be availeble

            public CheckBox UpdateCheckBox;
            public CheckBox DontShowThisUpdateAgainCheckBox;
            public Literal HiddenProductGUID;
            public Literal HiddenProductVersion;

            // ### UpdateCell Constructors
            //Description should contain size, downloadtime etc.
```

```
            public UpdateCell(string ProductGUID,  string ProductTitle,
List<UpdaterCommonLib.UpdateRequirement> Requirements, string Description, string Version,
Control Parent): base()
            {
                UpdateCheckBox = new CheckBox();
                UpdateCheckBox.ID = ProductGUID + "UpdateCheck";

                HtmlTable ProductTable = new HtmlTable();
                this.Controls.Add(ProductTable);

                HtmlTableRow TR = new HtmlTableRow();
                ProductTable.Rows.Add(TR);

                // 1st row
                HtmlTableCell TD = new HtmlTableCell();
                TR.Cells.Add(TD);
                TD.InnerHtml = "<img id='ShowHideImg_" + Id + "'
src='/KundeServer/images/gray1-plus.jpg' onclick='JavaScript:toggleDescription(" + Id + ")'
/>";


                TD = new HtmlTableCell();
                TR.Cells.Add(TD);
                TD.Controls.Add(UpdateCheckBox);
                UpdateCheckBox.Enabled = (Requirements.Count == 0);
                UpdateCheckBox.Checked = (Requirements.Count == 0);
                HiddenProductGUID = new Literal();
                HiddenProductGUID.Visible = false;
                HiddenProductGUID.Text = ProductGUID;
                TD.Controls.Add(HiddenProductGUID);

                HiddenProductVersion = new Literal();
                HiddenProductVersion.Visible = false;
                HiddenProductVersion.Text = Version;
                TD.Controls.Add(HiddenProductVersion);

                TD = new HtmlTableCell();
                TR.Cells.Add(TD);
                TD.InnerHtml = "<a href='JavaScript:toggleDescription(" + Id + ")'>" +
ProductTitle + " (" + Version + ")</a>";

                if (Requirements.Count > 0)
                {
                    string Reqs = "";
                    Reqs = "<br /><span style='font-weight:bold;'>Denne opgradering kan ikke
gennemføres, følgende produkter er påkrævet: ";
                    foreach (UpdateRequirement UR in Requirements)
                        Reqs += "<span style='color:red;'>" + UR.Name + " (" + UR.VersionNumber
+ ")</span>, ";
                    Reqs = Reqs.TrimEnd(',', ' ');
                    Reqs += "</span>";
                    TD.InnerHtml += Reqs;
                }

                // 2nd row
                TR = new HtmlTableRow();
                ProductTable.Rows.Add(TR);
                TR.ID = "DescriptionTR_" + Id;
                TR.Style[HtmlTextWriterStyle.Display] = "none";

                TD = new HtmlTableCell();
                TR.Cells.Add(TD);
                TD.ColSpan = 2;
                TD.InnerHtml = " ";

                TD = new HtmlTableCell();
                TR.Cells.Add(TD);
                TD.InnerHtml = Description + " ";

                // 3rd row
```

```
                TR = new HtmlTableRow();
                ProductTable.Rows.Add(TR);
                TR.ID = "DontShowTR_" + Id;
                TR.Style[HtmlTextWriterStyle.Display] = "none";

                TD = new HtmlTableCell();
                TR.Cells.Add(TD);
                TD.ColSpan = 2;
                TD.InnerHtml = " ";

                TD = new HtmlTableCell();
                TR.Cells.Add(TD);

                DontShowThisUpdateAgainCheckBox = new CheckBox();
                DontShowThisUpdateAgainCheckBox.ID = ProductGUID + "_DontShow";
                DontShowThisUpdateAgainCheckBox.Attributes.Add("onclick",
Parent.Page.ClientScript.GetPostBackEventReference(Parent, "HideProduct¤" + ProductGUID + "¤" +
ProductTitle, false));
                TD.Controls.Add(DontShowThisUpdateAgainCheckBox);
                DontShowThisUpdateAgainCheckBox.Text = "Vis ikke opgraderinger for dette
produkt igen";
                DontShowThisUpdateAgainCheckBox.TextAlign = TextAlign.Right;
                //DontShowThisUpdateAgainCheckBox.Enabled = false;

                Id++;
            }
        }

        private List<Product> InstalledProducts
        {
            get
            {
                if (Context.Items["InstalledProducts"] == null)
                {
                    List<Product> RtnList = new List<Product>();
                    XmlDocument XD = new XmlDocument();
                    XD.LoadXml(Functions.GetDataFromUpdater("GetInstalledProducts"));
                    foreach (XmlNode Node in XD.SelectNodes("/Products/VersionReference"))

                        RtnList.Add(new Product(null,
Functions.LoadXmlFromData(Functions.GetDataFromUpdater("GetProduct:" +
Node.SelectSingleNode("ProductGUID").InnerText))));
                    Context.Items["InstalledProducts"] = RtnList;
                }
                return (List<Product>)Context.Items["InstalledProducts"];
            }
        }
        private List<Update> AvailableUpdates
        {
            get
            {
                if (Context.Items["AvailableUpdates"] == null)
                {
                    List<Update> Updates = new List<Update>();
                    XmlDocument XD = new XmlDocument();
                    XD.LoadXml(Functions.GetDataFromUpdater("GetUpdates"));
                    foreach (XmlNode Node in XD.SelectNodes("/Updates/Update"))
                        Updates.Add(new Update(Node));
                    Context.Items["AvailableUpdates"] = Updates;
                }
                return (List<Update>)Context.Items["AvailableUpdates"];
            }
        }
        public Settings CurrentSettings
        {
            get
            {
                if (Context.Items["Settings"] == null)
                {
                    XmlDocument XD = new XmlDocument();
```

```
                    XD.LoadXml(Functions.GetDataFromUpdater("GetSettings"));
                    Context.Items["Settings"] = new Settings(XD);
                }

                return (Settings)Context.Items["Settings"];
            }
        }

        private List<string> Categories
        {
            get
            {
                List<string> RtnList = new List<string>();
                foreach (Update U in AvailableUpdates)
                    if (RtnList.Find(delegate(string S) { return S.Equals(U.CategoryName); })
== null)
                        RtnList.Add(U.CategoryName);
                return RtnList;
            }
        }
        private List<Update> GetUpdates(string Category)
        {
            return AvailableUpdates.FindAll(delegate(Update U) { return
U.CategoryName.Equals(Category); });
        }

        protected HtmlTable VersionTable = new HtmlTable();
        protected HtmlTable CustomerInfoTable = new HtmlTable();
        protected TextBox CompanyName;
        protected List<CheckBox> DisabledProducts = new List<CheckBox>();
        protected Button UpdateButton = new Button();
        protected Button CancelButton = new Button();
        protected Button OkButton = new Button();

        private string JSScript
        {
            get
            {
                return @"<script type='text/javascript'><!--
                    function toggleDescription(id){
                        var descriptionTR = document.getElementById('" +
this.UniqueID.Substring(0, this.UniqueID.IndexOf("$"))  + @"_DescriptionTR_'+id);
                        var dontShowTR = document.getElementById('" +
this.UniqueID.Substring(0, this.UniqueID.IndexOf("$")) + @"_DontShowTR_'+id);
                        var ShowHideImg = document.getElementById('ShowHideImg_'+id);
                        var show = (descriptionTR.style.display == 'none');

                        if(show){
                            descriptionTR.style.display = 'block';
                            dontShowTR.style.display = 'block';
                            ShowHideImg.src = '/KundeServer/images/gray1-minus.jpg';
                        }else{
                            descriptionTR.style.display = 'none';
                            dontShowTR.style.display = 'none';
                            ShowHideImg.src = '/KundeServer/images/gray1-plus.jpg';
                        }
                    }

                    function ShowHide(Show, Hide)
                    {
                        document.getElementById(Show).style.display = '';
                        document.getElementById(Hide).style.display = 'none';
                    }
                //-->
                </script>
                ";
            }
        }

        protected override void OnInit(EventArgs e)
```

```
        {
            if (Page.Request.Form["__EVENTTARGET"] != null &&
Page.Request.Form["__EVENTTARGET"].Equals(this.UniqueID))
            {
                if (Page.Request.Form["__EVENTARGUMENT"].StartsWith("HideProduct¤"))
                {
                    string ProductGUID = Page.Request.Form["__EVENTARGUMENT"].Split('¤')[1];
                    string ProductTitle = Page.Request.Form["__EVENTARGUMENT"].Split('¤')[2];
                    CurrentSettings.DisabledProducts.Add(new Settings.Product(ProductGUID,
ProductTitle));
                    Functions.GetDataFromUpdater("SaveSettings:" + CurrentSettings.ToXml(new
XmlDocument()).OuterXml);
                }
            }
            UpdateButton.Click += new EventHandler(UpdateButton_Click);
            EnsureChildControls();
        }

        protected override void CreateChildControls()
        {
            this.Controls.Add(VersionTable);
            this.Controls.Add(CustomerInfoTable);
            CreateUpdateForm();
            CreateCustomerInfoForm();

            if (CompanyName.Text.Length <= 0)
            {
                CustomerInfoTable.Style["Display"] = "";
                VersionTable.Style["Display"] = "None";
            }
        }
        protected void CreateUpdateForm()
        {
            //Register Show/Hide description script
            Page.ClientScript.RegisterClientScriptBlock(typeof(string),
"ClientControl_JSScript", JSScript);

            //Show table
            //this.Controls.Add(VersionTable);
            VersionTable.Attributes["class"] = "CC_Table";
            //VersionTable.Style["Display"] = "None";
            VersionTable.Border = 0;
            VersionTable.CellSpacing = 0;

            //Header row
            HtmlTableRow HeaderTR = new HtmlTableRow();
            VersionTable.Rows.Add(HeaderTR);
            HeaderTR.Attributes["class"] = "CC_HeaderTR";

            //HeaderCell HeaderTD = new HeaderCell("Tilgængelige opdateringer<br><a href=\"" +
Page.ClientScript.GetPostBackClientHyperlink(this, "Admin") + "\" style='float:right;font-
size:12px;font-weight:normal'>Administer</a> ");
            HeaderCell HeaderTD = new HeaderCell("Tilgængelige opdateringer<br><a
href=\"javascript:ShowHide('" + CustomerInfoTable.ClientID + "','" + VersionTable.ClientID +
"');\" style='float:right;font-size:12px;font-weight:normal'>Administrer</a> ");
            HeaderTR.Cells.Add(HeaderTD);

            //Content rows
            if (Categories.Count == 0)
            {
                HtmlTableRow CategoryTR = new HtmlTableRow();
                VersionTable.Rows.Add(CategoryTR);
                CategoryTR.Attributes["class"] = "CC_NoUpdatesTR";

                ContentCell CategoryTD = new ContentCell("Der er ikke nogle tilgængelige
opdateringer.");
                CategoryTR.Cells.Add(CategoryTD);
            }
            else
            {
```

```
                foreach (string Cat in Categories)
                {
                    HtmlTableRow CategoryTR = new HtmlTableRow();
                    VersionTable.Rows.Add(CategoryTR);
                    CategoryTR.Attributes["class"] = "CC_CategoryTR";

                    ContentCell CategoryTD = new ContentCell(Cat);
                    CategoryTR.Cells.Add(CategoryTD);

                    HtmlTableRow ContentTR;
                    UpdateCell ContentTD;
                    bool IsEvenRow = true;
                    foreach (Update U in GetUpdates(Cat))
                    {
                        // Check if product is disabled
                        if (CurrentSettings.DisabledProducts.Find(delegate(Settings.Product SP)
{ return SP.GUID.Equals(U.GUID); }) == null)
                        {
                            ContentTR = new HtmlTableRow();
                            VersionTable.Rows.Add(ContentTR);
                            ContentTR.Attributes["class"] = (IsEvenRow) ? "CC_EvenContentTR" :
"CC_OddContentTR";

                            //ContentTD = new UpdateCell(P.GUID.ToString(),
P.CurrentVersion.GUID.ToString(), P.Name + " " + P.CurrentVersion.Version,
KC.GetUnfufilledRequirements(CurrentCustomer, P), "Størrelse: " +
FileFunctions.GetSizeString(P.CurrentVersion.FileSize) + "<br />" +
P.CurrentVersion.Description, P.CurrentVersion.Version.ToString(), this);
                            ContentTD = new
UpdateCell(U.GUID,U.Name,U.Requirements,U.Description,U.VersionNumber.ToString(),this);
                            ContentTR.Cells.Add(ContentTD);

                            IsEvenRow = !IsEvenRow;
                        }
                    }
                }
            }

            HtmlTableRow UpdateTR = new HtmlTableRow();
            VersionTable.Rows.Add(UpdateTR);

            HtmlTableCell UpdateTD = new HtmlTableCell();
            UpdateTR.Cells.Add(UpdateTD);
            UpdateTD.Controls.Add(UpdateButton);
            UpdateButton.Text = "Opdater";
            UpdateButton.Enabled = (Categories.Count != 0);
        }
        protected void CreateCustomerInfoForm()
        {
            HtmlTableRow TR;
            HtmlTableCell TD;
            CustomerInfoTable.Attributes["class"] = "CC_Table";
            CustomerInfoTable.Style["Display"] = "None";
            //this.Controls.Add(CustomerInfoTable);

            // Header
            TR = new HtmlTableRow();
            TR.Attributes["class"] = "CC_HeaderTR";
            CustomerInfoTable.Rows.Add(TR);
            TD = new HeaderCell();
            TD.InnerHtml = "Administrer<br>";
            TD.ColSpan = 2;
            TR.Cells.Add(TD);

            // Companyname
            TR = new HtmlTableRow();
            CustomerInfoTable.Rows.Add(TR);
            TD = new HtmlTableCell();
            TD.InnerText = "Firmanavn:";
            TR.Cells.Add(TD);
```

```
            TD = new HtmlTableCell();
            CompanyName = new TextBox();
            CompanyName.Text = CurrentSettings.CompanyName;
            TD.Controls.Add(CompanyName);
            TR.Cells.Add(TD);

            // Deactivated products
            TR = new HtmlTableRow();
            TR.Attributes["class"] = "CC_CategoryTR";
            CustomerInfoTable.Rows.Add(TR);
            TD = new HtmlTableCell();
            TD.ColSpan = 2;
            TD.InnerText = "Deaktiveret produktopdateringer";
            TR.Cells.Add(TD);

            // products
            foreach (Product P in InstalledProducts)
            {
                TR = new HtmlTableRow();
                CustomerInfoTable.Rows.Add(TR);
                TD = new HtmlTableCell();
                TD.ColSpan = 2;
                CheckBox ProductCheckBox = new CheckBox();
                DisabledProducts.Add(ProductCheckBox);
                ProductCheckBox.Text = P.Name;
                ProductCheckBox.Attributes["GUID"] = P.GUID;
                ProductCheckBox.Checked =
CurrentSettings.DisabledProducts.Find(delegate(Settings.Product SP) { return
SP.GUID.Equals(P.GUID); }) != null;
                TD.Controls.Add(ProductCheckBox);
                TR.Cells.Add(TD);
            }

            TR = new HtmlTableRow();
            CustomerInfoTable.Rows.Add(TR);
            TD = new HtmlTableCell();
            TR.Cells.Add(TD);
            TD.ColSpan = 2;
            TD.Controls.Add(CancelButton);
            TD.Controls.Add(OkButton);

            OkButton.Text = "Ok";
            OkButton.Click += new EventHandler(OkButton_Click);
            OkButton.Width = new Unit("75px");

            CancelButton.Text = "Annuller";
            CancelButton.OnClientClick = "ShowHide('" + VersionTable.ClientID + "','" +
CustomerInfoTable.ClientID + "');return false;";
            CancelButton.Width = new Unit("75px");

            CompanyName.Attributes["onkeyup"] = "document.getElementById('" + OkButton.ClientID
+ "').disabled=this.value.length<=0";
            if (CompanyName.Text.Length <= 0)
            {
                OkButton.Attributes["disabled"] = "true";
                CancelButton.Attributes["disabled"] = "true";
            }
        }

        void OkButton_Click(object sender, EventArgs e)
        {
            CurrentSettings.CompanyName = CompanyName.Text;
            CurrentSettings.DisabledProducts.Clear();
            foreach (CheckBox CB in DisabledProducts)
            {
                if (CB.Checked)
                    CurrentSettings.DisabledProducts.Add(new
Settings.Product(CB.Attributes["GUID"], CB.Text));
            }
```

```
            Functions.GetDataFromUpdater("SaveSettings:" + CurrentSettings.ToXml(new
XmlDocument()).OuterXml);

            // Recreate control
            HttpContext.Current.Server.Transfer(Page.Request.Path);
        }

        protected void UpdateButton_Click(object sender, EventArgs e)
        {
            string ProcessGUID = ""; ;

            XmlDocument XD = new XmlDocument();
            XmlElement Jobs = XD.CreateElement("Jobs");
            XD.AppendChild(Jobs);

            foreach (HtmlTableRow TR in VersionTable.Rows)
            {
                foreach (HtmlTableCell TD in TR.Cells)
                {
                    if (TD is UpdateCell)
                    {
                        string ProductGUID = ((UpdateCell)TD).HiddenProductGUID.Text;
                        string ProductVersion = ((UpdateCell)TD).HiddenProductVersion.Text;
                        if (((UpdateCell)TD).UpdateCheckBox.Checked)
                        {
                            Jobs.AppendChild((new VersionReference(ProductGUID, new
System.Version(ProductVersion)).ToXml(XD)));
                        }
                    }
                }
            }
            //HttpContext.Current.Trace.Write(CBString);

            if (XD.ChildNodes.Count > 0)
            {
                ProcessGUID = Functions.GetDataFromUpdater("NewJob:" + XD.OuterXml);
            //    System.Net.Sockets.Socket S = new
System.Net.Sockets.Socket(System.Net.Sockets.AddressFamily.InterNetwork,
System.Net.Sockets.SocketType.Stream, System.Net.Sockets.ProtocolType.Tcp);

                ShowStatusModal(ProcessGUID);
            }
        }
        private void ShowStatusModal(string GUID)
        {
            Page.ClientScript.RegisterStartupScript(typeof(string), "installer",
"<script>window.showModalDialog('http://localhost:82/Status.htm?GUID=" + GUID +
"',null,'help:no;dialogWidth:850px;dialogHeight:440px;');</script>");

            Page.ClientScript.RegisterStartupScript(typeof(string), "refresh", "<script>" +
Page.ClientScript.GetPostBackEventReference(this, "UpgradeDone¤" + GUID) + "//window.location =
document.URL;</script>");
        }
    }
}
```

VersionsServer (Website)

VersionsService.cs

```
using System;
using System.Collections.Generic;
using System.Xml;
using System.Web.Services;
//using System.Web.Services.Protocols;
using AWO.CommonLib;
using AWO.UpdaterCommonLib;

public class VersionsService : System.Web.Services.WebService
{
    //### Private properties
    private string ProductsXmlFile
    {
        get {return Server.MapPath("~/App_Data/Products.xml");}
    }
    private string CustomersXmlFile
    {
        get {return Server.MapPath("~/App_Data/Customers.xml");}
    }
    private string CustomersLogFile(string CustomerGUID)
    {
        return Server.MapPath("~/App_Data/CustomerLog-" + CustomerGUID + ".xml");
    }
    private CategoryCollection Categories
    {
        get
        {
            if (Application["Categories"] == null)
            {
                Application.Lock();
                Application["Categories"] = new CategoryCollection();
                XmlDocument XD = new XmlDocument();
                XD.Load(ProductsXmlFile);
                ((CategoryCollection)Application["Categories"]).FromXml(XD);
                Application.UnLock();
            }
            return (CategoryCollection)Application["Categories"];
        }
    }

    //### Contructors
    public VersionsService()
    {
    }

    //### WebMethods
    [WebMethod] public string[] GetUpdates(string[] InstalledProducts)
    {
        List<VersionReference> InstalledVersions = new List<VersionReference>();
        foreach (string S in InstalledProducts)
            InstalledVersions.Add(new VersionReference(Functions.LoadXmlFromData(S)));

        List<string> AvailableUpdates = new List<string>(); // List to be returned

        foreach (VersionReference VR in InstalledVersions)
        {
            Product InstalledProduct = Categories.FindProduct(VR.ProductGUID);
            List<AWO.CommonLib.Version> NewerVersions =
InstalledProduct.GetVersionsNewerThan(VR.VersionNumber);
            if (NewerVersions.Count>0) // there's one or more newer versions. Find best match
            {
                NewerVersions.Sort();
                NewerVersions.Reverse();
```

```
                AWO.CommonLib.Version VersionToUse =
NewerVersions.FindLast(delegate(AWO.CommonLib.Version V) { return V.Active; });
                foreach (AWO.CommonLib.Version V in NewerVersions)
                {
                    // Check if requirements are ok
                    for (int i = V.RequiredVersions.Count - 1; i >= 0; i--) // Remove installed
requirements
                    {
                        VersionReference Requirement = V.RequiredVersions[i];
                        VersionReference InstalledVR =
InstalledVersions.Find(delegate(VersionReference VRi) { return
VRi.ProductGUID.Equals(Requirement.ProductGUID); });
                        if (InstalledVR.VersionNumber.CompareTo(Requirement.VersionNumber) != -
1) // installedversion>=requirement
                            V.RequiredVersions.RemoveAt(i);
                    }
                    if (V.RequiredVersions.Count == 0 && V.Active)
                    {
                        VersionToUse = V;
                        break;
                    }
                }

                AvailableUpdates.Add((new Update(VersionToUse, Categories)).ToXml(new
XmlDocument()).OuterXml);
                //AvailableUpdates.Add(VersionToUse.ToXml(new XmlDocument()).OuterXml);
            }
        }

        return AvailableUpdates.ToArray();
    }
    [WebMethod] public string GetCustomerLog()
    {
        XmlDocument XD = new XmlDocument();
        XD.Load(CustomersXmlFile);
        return XD.OuterXml;
    }
    [WebMethod] public void LogBeforeUpdate(string CustomerGUID, string ProcessGUID, string
InstalledVersionReferences, string UpdateVersionReferences)
    {
        // Extract lists from xml
        List<VersionReference> InstalledProducts = new List<VersionReference>();
        List<VersionReference> UpdateVersions = new List<VersionReference>();

        XmlElement InstalledProductsXE = Functions.LoadXmlFromData(InstalledVersionReferences);
        foreach (XmlNode N in
InstalledProductsXE.SelectNodes("/VersionReferences/VersionReference"))
            InstalledProducts.Add(new VersionReference(N));

        XmlElement UpdateVersionsXE = Functions.LoadXmlFromData(UpdateVersionReferences);
        foreach (XmlNode N in
UpdateVersionsXE.SelectNodes("/VersionReferences/VersionReference"))
            UpdateVersions.Add(new VersionReference(N));

        // Load log
        string FileName = CustomersLogFile(CustomerGUID);
        XmlDocument LogXD = new XmlDocument();
        if (System.IO.File.Exists(FileName))
            LogXD.Load(FileName);
        else
        {
            XmlElement XE = LogXD.CreateElement("Activities");
            LogXD.AppendChild(XE);
        }

        // Create new entry
        XmlElement Activities = (XmlElement)LogXD.SelectSingleNode("/Activities");

        XmlElement Activity = LogXD.CreateElement("Activity");
        Activities.AppendChild(Activity);
```

```
        XmlAttribute XA;

        XA = LogXD.CreateAttribute("Type");
        XA.Value = "Update";
        Activity.Attributes.Append(XA);

        XA = LogXD.CreateAttribute("ProcessGUID");
        XA.Value = ProcessGUID;
        Activity.Attributes.Append(XA);

        XA = LogXD.CreateAttribute("StartTime");
        XA.Value = DateTime.Now.ToString();
        Activity.Attributes.Append(XA);

        // add products to entry
        foreach (VersionReference VR in InstalledProducts)
        {
            Product P = Categories.FindProduct(VR.ProductGUID);
            XmlElement Product = LogXD.CreateElement("Product");
            Activity.AppendChild(Product);

            XA = LogXD.CreateAttribute("Name");
            XA.Value = P.Name;
            Product.Attributes.Append(XA);

            XA = LogXD.CreateAttribute("GUID");
            XA.Value = P.GUID;
            Product.Attributes.Append(XA);

            XA = LogXD.CreateAttribute("VersionBefore");
            XA.Value = VR.VersionNumber.ToString();
            Product.Attributes.Append(XA);

            //updateinfo
            VersionReference UpdateVR = UpdateVersions.Find(delegate(VersionReference VRi) {
return VRi.ProductGUID.Equals(P.GUID); });
            XA = LogXD.CreateAttribute("AttemptedUpdate");
            Product.Attributes.Append(XA);
            if (UpdateVR == null)
                XA.Value = false.ToString();
            else
            {
                AWO.CommonLib.Version UpdateVersion = P.FindVersion(UpdateVR.VersionNumber);

                XA.Value = true.ToString();
                XmlElement UpdateInfo = LogXD.CreateElement("UpdateInfo");
                Product.AppendChild(UpdateInfo);

                XA = LogXD.CreateAttribute("Filename");
                XA.Value = UpdateVersion.FilePath;
                UpdateInfo.Attributes.Append(XA);

                XA = LogXD.CreateAttribute("CommandArguments");
                XA.Value = UpdateVersion.CommandArgument;
                UpdateInfo.Attributes.Append(XA);

                XA = LogXD.CreateAttribute("Version");
                XA.Value = UpdateVersion.VersionNumber.ToString();
                UpdateInfo.Attributes.Append(XA);
            }
        }
        System.IO.File.SetAttributes(FileName, System.IO.FileAttributes.Normal);
        LogXD.Save(FileName);
    }
    [WebMethod] public void LogAfterUpdate(string CustomerGUID, string ProcessGUID, string
InstalledVersionReferences)
    {
        // Extract lists from xml
        List<VersionReference> InstalledProducts = new List<VersionReference>();
```

```
        XmlElement InstalledProductsXE = Functions.LoadXmlFromData(InstalledVersionReferences);
        foreach (XmlNode N in
InstalledProductsXE.SelectNodes("/VersionReferences/VersionReference"))
            InstalledProducts.Add(new VersionReference(N));

        // Load log
        string FileName = CustomersLogFile(CustomerGUID);
        XmlDocument LogXD = new XmlDocument();
        LogXD.Load(FileName);

        // Find update Process, for update
        XmlElement Node = null;
        foreach (XmlNode N in LogXD.SelectNodes("/Activities/Activity"))
            if (N.Attributes["ProcessGUID"].Value.Equals(ProcessGUID))
                Node = (XmlElement)N;

        // Update elements
        XmlAttribute EndTime = LogXD.CreateAttribute("EndTime");
        EndTime.Value = DateTime.Now.ToString();
        Node.Attributes.Append(EndTime);

        // Update products
        foreach (XmlNode N in Node.SelectNodes("Product"))
        {
            System.Version V = InstalledProducts.Find(delegate(VersionReference VR) { return
VR.ProductGUID.Equals(N.Attributes["GUID"].Value); }).VersionNumber;
            XmlAttribute VersionAfter = LogXD.CreateAttribute("VersionAfter");
            VersionAfter.Value = V.ToString();
            N.Attributes.Append(VersionAfter);
        }
        LogXD.Save(FileName);

        // ### Update customer
        if(!System.IO.File.Exists(CustomersXmlFile))
        {
            LogXD = new XmlDocument();
            XmlElement XE = LogXD.CreateElement("Customers");
            LogXD.AppendChild(XE);
        }
        else
            LogXD.Load(CustomersXmlFile);

        //Find CustomerNode
        XmlElement CustomerNode = null;
        foreach (XmlNode N in LogXD.SelectNodes("/Customers/Customer"))
            if (N.Attributes["GUID"].Value.Equals(CustomerGUID))
                CustomerNode = (XmlElement)N;
        if (CustomerNode == null)
        {
            XmlElement XE = LogXD.CreateElement("Customer");
            LogXD.FirstChild.AppendChild(XE);
            CustomerNode = XE;

            XmlAttribute Name = LogXD.CreateAttribute("Name");
            CustomerNode.Attributes.Append(Name);

            XmlAttribute GUID = LogXD.CreateAttribute("GUID");
            GUID.Value = CustomerGUID;
            CustomerNode.Attributes.Append(GUID);
        }

        XmlElement InstalledProductsNode =
(XmlElement)CustomerNode.SelectSingleNode("InstalledProducts");
        if (InstalledProductsNode != null)
            InstalledProductsNode.RemoveAll();
        else
        {
            InstalledProductsNode = LogXD.CreateElement("InstalledProducts");
            CustomerNode.AppendChild(InstalledProductsNode);
```

```
        }
        XmlAttribute LastUpdated = LogXD.CreateAttribute("LastUpdated");
        LastUpdated.Value = DateTime.Now.ToString();
        InstalledProductsNode.Attributes.Append(LastUpdated);
        foreach (VersionReference VR in InstalledProducts)
            InstalledProductsNode.AppendChild(VR.ToXml(LogXD));
        System.IO.File.SetAttributes(CustomersXmlFile, System.IO.FileAttributes.Normal);
        LogXD.Save(CustomersXmlFile);
    }
    [WebMethod] public void UpdateCompanyName(string CustomerGUID, string CompanyName)
    {
        XmlDocument LogXD = new XmlDocument();
        if (!System.IO.File.Exists(CustomersXmlFile))
        {
            XmlElement XE = LogXD.CreateElement("Customers");
            LogXD.AppendChild(XE);
        }
        else
            LogXD.Load(CustomersXmlFile);

        //Find CustomerNode
        XmlElement CustomerNode = null;
        foreach (XmlNode N in LogXD.SelectNodes("/Customers/Customer"))
            if (N.Attributes["GUID"].Value.Equals(CustomerGUID))
                CustomerNode = (XmlElement)N;
        if (CustomerNode == null)
        {
            XmlElement XE = LogXD.CreateElement("Customer");
            LogXD.FirstChild.AppendChild(XE);
            CustomerNode = XE;

            XmlAttribute Name = LogXD.CreateAttribute("Name");
            Name.Value = CompanyName;
            CustomerNode.Attributes.Append(Name);

            XmlAttribute GUID = LogXD.CreateAttribute("GUID");
            GUID.Value = CustomerGUID;
            CustomerNode.Attributes.Append(GUID);
        }
        else
            CustomerNode.Attributes["Name"].Value = CompanyName;
        System.IO.File.SetAttributes(CustomersXmlFile, System.IO.FileAttributes.Normal);
        LogXD.Save(CustomersXmlFile);
    }
    //Category
    [WebMethod] public string GetCategories()
    {
        return Categories.ToXml(new XmlDocument()).OuterXml;
    }
    [WebMethod] public void AddEditCategory(string CategoryData)
    {
        if (CheckForAdminRights())
        {
            Category C = new Category(Categories, Functions.LoadXmlFromData(CategoryData));
            Category CategoryToUpdate = Categories.FindCategory(C.GUID);
            if (CategoryToUpdate == null)
            {
                CategoryToUpdate = C;
                CategoryToUpdate.GUID = Guid.NewGuid().ToString();
                Categories.Add(C);
            }
            else
                CategoryToUpdate.FromXml(Functions.LoadXmlFromData(CategoryData));
            SaveProducts();
        }
    }
    [WebMethod] public void DeleteCategory(string GUID)
    {
        if (CheckForAdminRights())
        {
```

```
            Categories.RemoveAll(delegate(Category C) { return C.GUID.Equals(GUID); });
            SaveProducts();
        }
    }
    //Product
    [WebMethod] public string[] GetAllProductGUIDs()
    {
        List<string> GUIDS = new List<string>();
        foreach (Category Cat in Categories)
            foreach (Product Pro in Cat)
                GUIDS.Add(Pro.GUID);
        return GUIDS.ToArray();
    }
    [WebMethod] public string GetProduct(string GUID)
    {
        return Categories.FindProduct(GUID).ToXml(new XmlDocument()).OuterXml;
    }
    [WebMethod] public void AddEditProduct(string GUID, string CategoryGUID, string
ProductData)
    {
        if (CheckForAdminRights())
        {
            Product P = new Product(Categories.FindCategory(CategoryGUID),
Functions.LoadXmlFromData(ProductData));
            Product ProductToUpdate = Categories.FindProduct(GUID);
            if (ProductToUpdate == null)
            {
                ProductToUpdate = P;
                ProductToUpdate.ParentCategory.Add(ProductToUpdate);
            }
            else
                ProductToUpdate.FromXml(Functions.LoadXmlFromData(ProductData));
            SaveProducts();
        }
    }
    [WebMethod] public void DeleteProduct(string GUID)
    {
        if (CheckForAdminRights())
        {
            Categories.FindProduct(GUID).ParentCategory.RemoveProduct(GUID);
            SaveProducts();
        }
    }
    //Version
    [WebMethod] public string[] GetJobs(string[] VersionReferences)
    {
        List<VersionReference> VRS = new List<VersionReference>();
        foreach (string S in VersionReferences)
            VRS.Add(new VersionReference(Functions.LoadXmlFromData(S)));

        //List<AWO.CommonLib.Version> Versions = new List<AWO.CommonLib.Version>();
        List<string> RtnList = new List<string>();
        foreach (VersionReference VR in VRS)
        {
            AWO.CommonLib.Version Ver = Categories.FindVersion(VR.ProductGUID,
VR.VersionNumber);
            RtnList.Add((new
Job(VR.ProductGUID,Categories.FindProduct(VR.ProductGUID).Name,VR.VersionNumber.ToString(),Ver.
FilePath,Functions.GetSizeString(Ver.FileSize),Ver.CommandArgument)).ToXml(new
XmlDocument()).OuterXml);
        }

        return RtnList.ToArray();
    }
    [WebMethod] public void AddEditVersion(string GUID, string ProductGUID, string VersionData)
    {
        if (CheckForAdminRights())
        {
```

```
            AWO.CommonLib.Version V = new
AWO.CommonLib.Version(Categories.FindProduct(ProductGUID),
Functions.LoadXmlFromData(VersionData));
            AWO.CommonLib.Version VersionToUpdate = Categories.FindVersion(GUID);
            if (VersionToUpdate == null)
            {
                VersionToUpdate = V;
                VersionToUpdate.GUID = Guid.NewGuid().ToString();
                V.Product.Add(V);
            }
            else
                VersionToUpdate.FromXml(Functions.LoadXmlFromData(VersionData));
            SaveProducts();
        }
    }
    [WebMethod] public void DeleteVersion(string GUID)
    {
        if (CheckForAdminRights())
        {
            Categories.FindVersion(GUID).Product.RemoveVersion(GUID);
            SaveProducts();
        }
    }
    //File
    [WebMethod] public string UploadFile(Byte[] Data, string ProductGUID, string Version,
string Filename)
    {
        if (CheckForAdminRights())
        {
            string DataRoot = Server.MapPath("~/Data");
            if (!System.IO.Directory.Exists(DataRoot + "\\" + ProductGUID))
                System.IO.Directory.CreateDirectory(DataRoot + "\\" + ProductGUID);
            System.IO.File.WriteAllBytes(DataRoot + "\\" + ProductGUID + "\\" + Version + "-" +
Filename, Data);

            string VDir =
System.Web.HttpContext.Current.Request.ServerVariables["SCRIPT_NAME"].Substring(0,
System.Web.HttpContext.Current.Request.ServerVariables["SCRIPT_NAME"].IndexOf("/", 2));
            return "http://" +
System.Web.HttpContext.Current.Request.ServerVariables["SERVER_NAME"] + VDir + "/Data/" +
ProductGUID + "/" + Version + "-" + Filename;
        }
        else
            return "";
    }
    //### Private helper methods
    private void SaveProducts()
    {
        XmlDocument XD = new XmlDocument();
        XD.AppendChild(Categories.ToXml(XD));
        System.IO.File.SetAttributes(ProductsXmlFile, System.IO.FileAttributes.Normal);
        XD.Save(ProductsXmlFile);
    }

    private bool CheckForAdminRights()
    {
        string RemoteIp = System.Web.HttpContext.Current.Request.UserHostAddress;
        foreach (string S in
System.Configuration.ConfigurationSettings.AppSettings.Get("AdminIps").Split(';'))
            if (S.Equals(RemoteIp))
                return true;
        return false;
    }
}
```

WebtopServer (Website)

AdminControlBase.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Xml;
using AWO.CommonLib;

public class AdminControlBase : UserControl
{
    private VersionServer.VersionsService _VS;
    protected VersionServer.VersionsService VS
    {
        get
        {
            if (_VS == null)
            {
                _VS = new VersionServer.VersionsService();
                _VS.Credentials = System.Net.CredentialCache.DefaultCredentials;
            }
            return _VS;
        }
    }
    private CategoryCollection _PC;
    protected CategoryCollection PC
    {
        get
        {
            if (_PC == null)
            {
                _PC = new CategoryCollection();
                _PC.FromXml(Functions.LoadXmlFromData(VS.GetCategories()));
            }
            return _PC;
        }
    }
}
```

Panel.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

namespace AWO.VersionsServer.Webcontrols
{
    public class Panel : PlaceHolder
    {
        public event EventHandler Close;

        public int Width = 300;
        public int Height = 150;
```

```
        public string Title = "Title her";
        public bool AutoHideAfterPostback = false;
        public bool ShowOkButton = false;
        private string ShowContext = "";
        public void Show(string Context)
        {
            this.Visible = true;
            ShowContext = "<center>" + Context + "</center>";
        }
        public void Show(string Context, int Width, int Height)
        {
            Show(Context);
            this.Width = Width;
            this.Height = Height;
        }

        protected override void OnLoad(EventArgs e)
        {
            if (Page.Request.Form["__EVENTTARGET"] != null &&
Page.Request.Form["__EVENTTARGET"].Equals(this.UniqueID))
            {
                if (Page.Request.Form["__EVENTARGUMENT"].Equals("Close"))
                {
                    if (Close != null)
                        Close(this, EventArgs.Empty);

                    //this.Visible = false;
                }
            }
            if (AutoHideAfterPostback)
                this.Visible = false;
        }
        protected override void Render(HtmlTextWriter writer)
        {
            string Top = "expression((document.body.clientHeight-" + Height + ")/2)";
            string Left = "expression((document.body.clientWidth-" + Width + ")/2)";

            string ShadowTop = "expression((document.body.clientHeight-" + Height + ")/2+4)";
            string ShadowLeft = "expression((document.body.clientWidth-" + Width + ")/2+4)";

            string CloseLink = "<a style='color:#000000;text-decoration:none;' href=\"" +
Page.ClientScript.GetPostBackClientHyperlink(this, "Close") + "\">[x]</a>";

            writer.WriteLine("<div style='background-
color:#FFFFFF;filter:progid:DXImageTransform.Microsoft.Alpha(opacity=30);position:absolute;top:
0px;left:0px;height:expression(document.body.clientHeight);width:expression(document.body.clien
tWidth);' ></div>");

            writer.WriteLine("<div style='background-color:#000000;position:absolute;top:" +
ShadowTop + ";left:" + ShadowLeft + ";height:" + Height + ";width:" + Width + ";' ></div>");

            writer.WriteLine("<table cellspacing=0 cellpadding=0 border=0
style='position:absolute;top:" + Top + ";left:" + Left + ";background-color:#FFFFFF;width:" +
Width + "px;height:" + Height + "px;border:solid 1px #000000;'>");
            writer.WriteLine("<tr><td style='Font-weight:bold;Height:20px;'> " + Title +
"</td><td style='Font-weight:bold;' align=right>" + CloseLink + " </td></tr>");

            writer.WriteLine("<tr><td colspan=2 style='border-top:solid 1px
#000000;padding:5px;height:" + (Height - 20) + "px;'>");
            base.Render(writer);
            if (ShowContext.Length > 0)
                writer.WriteLine(ShowContext);
            if (ShowOkButton)
            {
                Button OkBtn = new Button();
                OkBtn.Text = "Ok";
                OkBtn.Width = new Unit(50);
                writer.WriteLine("<center>");
                OkBtn.RenderControl(writer);
                writer.WriteLine("</center>");
```

```
            }
            writer.WriteLine("</td></tr>");
            writer.WriteLine("</table>");

            // Clear ShowContext for next postback
            ShowContext = "";
        }
    }
}
```

TabView.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;


namespace AWO.VersionsServer.Webcontrols
{
    public class TabView : MultiView
    {
        // ### Public properties
        public string BorderWidth
        {
            get { return _BorderWidth; }
            set { _BorderWidth = value; }
        }
        public string BorderColor
        {
            get { return _BorderColor; }
            set { _BorderColor = value; }
        }
        public string Width
        {
            get { return _Width; }
            set { _Width = value; }
        }
        public string TabSpacerWidth
        {
            get { return _TabSpacerWidth; }
            set { _TabSpacerWidth = value; }
        }

        public string ActiveTabCSS
        {
            get { return _ActiveTabCSS; }
            set { _ActiveTabCSS = value; }
        }
        public string TabCSS
        {
            get { return _TabCSS; }
            set { _TabCSS = value; }
        }
        public string TabHoverCSS
        {
            get { return _TabHoverCSS; }
            set { _TabHoverCSS = value; }
        }
        public string ContentCSS
        {
            get { return _ContentCSS; }
            set { _ContentCSS = value; }
        }
```

```
        // ### Private properties
        private string _BorderWidth = "1px";
        private string _BorderColor = "#000000";
        private string _Width = "800px";
        private string _TabSpacerWidth = "30px";
        private string _ActiveTabCSS = "";
        private string _TabCSS = "";
        private string _TabHoverCSS = "";
        private string _ContentCSS = "";
        private string JavaScript
        {
            get
            {
                return @"
<script>
    function TabView_TabOver(td, CssClass)
    {
        td.className = CssClass;
    }
    function TabView_TabOut(td, CssClass)
    {
        td.className = CssClass;
    }
</script>
                ";
            }
        }

        // ### Overrides
        protected override void OnLoad(EventArgs e)
        {
            if (Page.Request.Form["__EVENTTARGET"] != null &&
Page.Request.Form["__EVENTTARGET"].Equals(this.UniqueID))
                ActiveViewIndex = int.Parse(Page.Request.Form["__EVENTARGUMENT"]);
            Page.ClientScript.RegisterClientScriptBlock(typeof(string), "TabView", JavaScript);
        }
        protected override void Render(HtmlTextWriter writer)
        {
            // Outer div... for size
            writer.WriteLine("<div style='width:" + Width + ";'>");

            // Tab table
            writer.WriteLine("<table cellspacing='0' cellpadding='0' border='0'
style='width:100%;border-collapse:collapse;'><tr>");
            for (int i = 0; i < Views.Count; i++)
            {
                // Tab
                string Link = Page.ClientScript.GetPostBackEventReference(this, i.ToString());
                string TabWidth = ((AWO.VersionsServer.Webcontrols.View)Views[i]).Width;
                if(i == ActiveViewIndex)
                    writer.WriteLine("<td onclick=\"" + Link + "\" class='" + ActiveTabCSS + "'
style='text-align:center;cursor:hand;border:solid #000000 1px;border-bottom:none;width:" +
TabWidth + ";'>" + ((AWO.VersionsServer.Webcontrols.View)Views[i]).Title + "</td>");
                else
                    writer.WriteLine("<td onclick=\"" + Link + "\" class='" + TabCSS + "'
style='text-align:center;cursor:hand;border:solid #000000 1px;width:" + TabWidth + ";'
onmouseover=\"TabView_TabOver(this, '" + TabHoverCSS + "');\" onmouseout=\"TabView_TabOut(this,
'" + TabCSS + "');\">" + ((AWO.VersionsServer.Webcontrols.View)Views[i]).Title + "</td>");

                // Spacer
                if(i<Views.Count-1)
                    writer.WriteLine("<td style='border-bottom:solid #000000 1px;width:" +
TabSpacerWidth + ";'> </td>");
                else
                    writer.WriteLine("<td style='border-bottom:solid #000000
1px;'> </td>");
            }
            writer.WriteLine("</tr></table>");
```

```
            // Content Table
            writer.WriteLine("<table cellspacing='0' cellpadding='0' border='0'
style='width:100%;border-collapse:collapse;'><tr><td class='" + ContentCSS + "'
style='border:solid #000000 1px;border-top:none;'>");
            base.Render(writer);
            writer.WriteLine("</table></tr></table>");

            writer.WriteLine("</div>");
        }
    }
}
```

View.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

namespace AWO.VersionsServer.Webcontrols
{
    public class View : System.Web.UI.WebControls.View
    {
        private string _Title;
        private string _Width = "150px";

        public string Width
        {
            get { return _Width; }
            set { _Width = value; }
        }

        public string Title
        {
            get { return _Title; }
            set { _Title = value; }
        }

    }
}
```