tordivel as
software solutions

tordivel as
storgata 20
N-0184 oslo
NORWAY
www.tordivel.com
www.tordivel.no
office@tordivel.no

Bankterminalkobling Programmeringsmanual

Versjon 2.1
1 desember 2005

OLE Automation programmeringsmanual for Bankterminalkobling, en Automation-komponent for kommunikasjon med betalingsterminaler ved hjelp av Bank Axepts SOFIE protokoll under Windows 32 bit plattform, Windows 95/98, Windows NT, 2000 eller XP.

Innhold

Innledning

Idag er de fleste varer og tjenester mulig å betale med bank- og kredittkort av forskjellige typer, kontant betaling blir etterhvert mindre og mindre brukt. Dette gjør at de fleste butikker har betalingsterminaler som er koplet opp mot Bankenes Betalings Sentral, BBS, og erstatter kontantoppgjør. Etterhvert har det også blitt vanlig at kunden kan ta ut mindre beløp i tillegg til å betale varer slik at handelsstedet også fungerer som bank/minibank.

Disse betalingsterminalene har et serielt grensesnitt med Bank Axepts SOFIE protokoll som annet utstyr kan tilkoples for å utføre funksjoner mot BBS. Dette kan typisk være kassesystemer i dagligvare- og faghandelen som ønsker automatisk kopling mot Bankenes Betalings Sentral for å slippe manuelt innslag av beløp, noe som kan være en kilde til feilinnslag og kontobelastning.

TORDIVEL AS Bankterminalkobling er en Windows 32 bit applikasjon som implementerer denne protokollen og tilbyr et enkelt OLE Automation grensesnitt. Dette grensenittet kan benyttes av de fleste utviklingsverktøy for Windows 32 bit plattformen. Leverandører av kasse- og betalingssystemer kan lett integrere dette i sine applikasjoner og få en feil- og vedlikeholdsfri kopling mot betalingsterminalen istedet for selv å implementere og teste SOFIE protokollen.

Dokumentreferanser

BBS AS, SOFIE, Chapter 13, ver-7.06, 25.02.2002
Installasjonsmanual for Bankterminalkobling, Tordivel AS, TD98002, 1999

Dokumentrevisjon

Revisjon	Endring
Td98001g 19aug2005	• Endringer o Python integrasjon
Td98001f 05des2003	• Endringer o SendAdmin - Adminkoder er hex-kodet istedet for desimal-kodet o PreAuthorize – signatur endret o Adjust – signatur endret • Utvidet grensenitt o GetSequenceNo
Td98001e 19mar2003	• Utvidet grensesnitt – nye funksjoner o PreAuthorize o Adjust o SendAdminCode o SendCharacter o SendCardData o SetOperId

Begreper

ECR	-	Electronica Cash Register
ITU	-	Integrated Terminal Unit
PAN	-	Primary Account Number
Transaksjon kort/konto	-	overføring av beløp, normalt en belastning av en kundes bank
Bank Mode	-	ECR's tilstand under en transaksjon/dialog med terminalen
Local Mode	-	ECR's tilstand når den ikke er i Bank Mode (idle)
Betalings system	-	en administrasjons applikasjon som styrer hele betalingssystemet
WordBool	-	2 bytes returverdi hvor 0 er False, alt annet er True.
Integer	-	4 bytes heltalls verdi (32 bit plattform)
WideString	-	en OLE Unicode null terminert karakter streng, 16 bits karakter sett.

Installasjon

Bankterminalkobling leveres med et eget installasjonssystem i to varianter, kasseinstallasjon og SDK installasjon. Kasseinstallasjonen installerer kun komponentene som er nødvendig for å kjøre serveren, SDK varianten installerer alle nødvendige komponenter for å kjøre serveren i tillegg til dokumentasjon og bruker eksempler i forskjellige utviklingsverktøy. For å ta i bruk serveren må den startes og konfigureres i konfigureringsvinduet. Deretter er serveren klar til bruk gjennom OLE Automation grensenittet. **Se installasjonsmanualen for ytterligere installasjons- og konfigurasjonsdetaljer.**

SOFIE protokollen

SOFIE protokollen er definert av Bank Axept AS (se dokumentreferanse). Dette dokumentet omhandler terminalens (ITU) og kassesystemets (ECR) virkemåte, en del tidskriterier, og hvordan de forskjellige meldinger skal settes opp og besvares. Alt dette tas hånd om av Bankterminalkobling, men en bør likevel ha en viss forståelse av hvordan en transaksjon foregår når en skal benytte denne serveren.

Local Mode / Bank Mode

Normalt er ECR i Local Mode, dvs. ECR er master i systemet og initierer alle dialoger mellom ITU og ECR. ITU er i denne tilstanden passiv (selv om den sender en jevn strøm av karakterer til ECR for å tilkjennegi sin eksistens). Men etter at ECR har initiert en transaksjon går tilstanden over i Bank Mode, ITU blir master og ECR blir passiv inntil ITU sender en Local Mode melding igjen etter at transaksjonen er ferdig. Dersom ITU ikke svarer, eller transaksjonen stopper opp underveis, vil etter en gitt tid (typisk 1 minutt) ECR gå over i Local Mode igjen. Dersom en transaksjon tar uventet lang tid kan ITU sende 'vent' kommandoer til ECR slik at ECR ikke automatisk går over i Local Mode.

Tekstmeldinger

I Bank Mode tilstand sender ITU en del tekstmeldinger som er ment å gå til enten en lokal skriver eller et lokalt informasjonsvindu. I dagligvare handelen er den lokale skriver normalt kvitteringsskriveren som skriver ut de artikler kunden skal belastes for, brød og melk. Disse skrivermeldingene skal normalt skrives ut på samme kvittering og må tas hånd om av systemapplikasjonen. Lokalvindusmeldinger er normalt samme enhet som viser hvilke varer som går forbi strekkodeleseren og totalsummen kunden skal belastes for. Begge disse enhetene kontrolleres fullt og helt av systemapplikasjonen.

Tekstmeldingene som genereres av ITU er tilgjengelig gjennom Bankterminalkoblingen. Hver slik melding skal imidlertid kvitteres av Bankterminalkoblingen, men denne har ikke kontroll

over de lokale enhetene og vet derfor ikke status for dem, ok, opptatt eller feil. Det er systemapplikasjonens ansvar å sette status for disse enheter ved funskjoner til Bankterminalkoblingen.

Dersom en skrivermelding under en transaksjon ikke blir kvittert med ok vil terminalen skrive ut en komplett kvittering på den interne kvitteringsskriveren (dersom terminalen har en slik). På enkelte terminaltyper får en heller ikke gjennomført en transaksjon dersom en ikke svarer lokalvindu ok på lokalvindumeldinger. *Ved oppstart av Bankterminalkobling er initiell verdi for både lokalskriver og lokalvindu enhetsfeil, denne må derfor endres av betalingssystemet*.

FIFO tekstbuffer

Alle skriver og lokalvindu tekstmeldinger blir lagt i FIFO buffer, First In First Out, et for lokalvindu- og et for skrivermeldinger. Der blir de liggende inntil systemapplikasjonen henter de ut *eller* en ny transaksjon blir initiert. Når systemapplikasjonen ber om en lokalvindu- eller skrivermelding, returneres alltid den eldste meldingen, og den blir slettet fra bufferet. Det er systemapplikasjonens ansvar å ta hånd om alle meldinger i disse bufferne før, under og etter en transaksjon. Bankterminalkobling har funksjoner for å returnere antall meldinger i bufferne og den meldingen som til enhver tid liggger på toppen, hvis noen. **Uavhentede meldinger som ligger i disse bufferne blir slettet ved initiering av en ny transaksjon uten varsel.**

Spesialkarakterer

Tekstmeldinger kan inneholde karakterer som må oversettes til den spesifikke kvitteringskriver. Det er kasseapplikasjonen som må oversette disse karakterene.

Karakter (HEX)	Oversettelse
5A	Æ
5B	Ø
5C	Å
7A	æ
7B	ø
7C	å
0A	Linjskift, formattering
0C	EOT, indikerer slutt på melding
0E	Papirkutt, skal alltid etterfølges av 0C (EOT)

Funksjonsgrensesnitt

Bankterminalkobling kan aksesseres ved hjelp av OLE Automation. Navnet på automation objektet er **BankAxeptSrv.BankAxeptAutomation**.

Hent status for tilkoplet terminal

Denne funksjonen returnerer status for kommunikasjonslinken mellom Bankterminalkobling og terminalen. Terminalen sender alltid karakter hex-05 for å fortelle serveren at den er tilstede og kommunikasjonen er intakt. Systemapplikasjonen bør sjekke om terminalen er tilkoplet før den prøver å kjøre noen transaksjoner.

function Connected : WordBool;

Returverdi	True for tilkoplet, False for kommunikasjons feil

Hent lisensstatus

Denne funksjonen returnerer om Bankterminalkobling er konfigurert rett. Under hver transaksjon sjekkes inntastet terminalnummer mot det unike terminalnummer til terminalen som er tilkoplet. Dersom terminalnummer ikke stemmer, eller lisensnummer som er lagt inn ikke stemmer med terminal nummeret returnerer denne funksjonen lisens ikke ok.

function LicenseVerified : WordBool;

Returverdi	True for ok, False for ikke ok.

Hent tilstand (Mode)

Denne funksjonen returner tilstand for ECR, Bank Mode eller Local Mode. Systemapplikasjonen bør sjekke at tilstand er Local Mode før den prøver å kjøre en transaksjon.

function BankMode : WordBool;

Returverdi	True for BankMode, False for LocalMode

Start transaksjon

Denne funksjonen starter en transaksjon, normalt en belastning på kundens bankkort. Tilstanden for ECR må være Local Mode før funksjonen utføres. Tilstanden går over til BankMode og terminalen styrer resten av transaksjonen når denne funksjonen kalles. Kunden blir bedt om å taste inn sin pin kode og terminalen ringer opp BBS, sjekker kortet og eventuelt utfører transaksjonen. Mens transaksjonen er under utførelse kommer skriver- og lokalvindus- meldinger som systemapplikasjonen må ta hånd om. **Dersom terminalnummer eller lisensnummer ikke er gyldig vil ikke denne funksjonen utføres.**

Denne funksjonen finnes i to varianter fra og med Bankterminalkobling versjon 1.3. Protokollen SOFIE 5 åpner for et nytt datafelt som kan sendes BBS ved transaksjon. Dette datafeltet, HostData parameteren, inneholder meldinger til kortselskap med fordelsavtaler, bonuskort, og er primært tiltenkt avanserte bonusberegninger. Dersom kunden benytter et fordelskort for dette brukerstedet, vil ikke dette datafeltet bli behandlet av BBS men istedet videresent til kortselskapet. Kortselskapet vil normalt sende et tilsvarende datafelt i retur til BBS, som igjen returnerer dette til terminalen ved transaksjonens slutt. Formatet på dette ekstra datafeltet er i utgangspunktet fritt for hvert enkelt kortselskap, men visse retningslinjer

vil likevel bli trukket av Bank Axept. For ytterligere informasjon om dette feltet må Bank Axept og aktuelt kortselskap kontaktes.

function TransferAmount (purchase, cashBack : Integer) : WordBool;

function TransferAmountEx (purchase, cashBack : Integer; hostdata : WideString) : WordBool;

purchase	beløp varekjøp, varer/tjenester kunden skal betale, **beløp i øre** (1/100 NOK).
cachback	kontant uttak, beløp kunden ønsker å ta ut i tillegg til varekjøp, **beløp i øre** (1/100 NOK).
hostdata	inntil 40 lesbare ASCII karakterer i området H20..H7F. Dette feltet kan også være tomt/blankt
Returverdi	True dersom dialog med terminalen er initiert. Dersom kommunikasjonsfeil, lisensfeil eller ECR allerede er i BankMode returneres False

***TransferAmountEx* kan kun benyttes dersom Bankterminalkobling er konfigurert til å bruke SOFIE 5 hostdata format, se konfigureringsmanual.**

Etterhvert er det blitt vanlig at kunden kan ta ut kontanter i tillegg til varekjøp. Rent kontantuttak vil også være mulig i enkelte systemer. Total summen som kunden skal belastes vil alltid være (purchase + cashBack). **Merk at purchase ikke er totalsummen kunden skal belasted, men kun de varer/tjenester det skal betales for.**

Kun kontantuttak, transaksjon uten varekjøp, gjøres ved å sette purchase til 0 og cashback til uttaksbeløp.

Fylling av kundekort (innskudd) gjøres ved å sette purchase til 0 og cashback til summen som skal settes inn på kortet men med negativt fortegn.

Saldoforespørsel gjøres ved å sette både purchase og cashback til 0.

Annulering av siste transaksjon er mulig ved å sette purchase lik summen av forrige purchase og cashback med negativt fortegn og cashback lik 0. Denne funksjonen krever at kasseoperatøren har et kjøpmannskort, dvs. et service/administrasjonskort. Kasseoperatøren må dra **kjøpmannskortet gjennom kortleseren** før annuleringstransaksjonen blir kjørt mot BBS. Denne funksjonen er normalt ikke kjent, det er heller ikke vanlig at kasseoperatøren har kjøpmannskort tilgjengelig. Annulering blir ikke godkjent dersom beløp ikke stemmer overens med siste transaksjon, delannulering er ikke mulig.

Hent transaksjonsresultat

Funksjon for å hente resultat for siste transaksjon.

function TransactionOk : WordBool;

Returverdi	True dersom transaksjonen ble godkjent av BBS, False dersom kortfeil, ikke dekning i kortet, transaksjon avbrutt eller av annen årsak ble avvist av BBS

Hent korttype

Denne funksjonen returnerer en tallkode som identifiserer korttype som ble brukt ved siste transaksjon. Denne informsjonen er av enkelte brukt til å føre statistikk og regnskap over de forskjellige kort som benyttes. Tallkoden som returneres er definert av Bank Axept og er en løpende liste over de kort som har avtaler med Bank Axept/BBS. Hver korttype vil få sin unike tallkode tildelt av Bank Axept. Se tabellen under for gyldige koder.

Kort ID	Kortselskap
01	Norske bankkort
02	Postbanken
03	Visa
04	Eurocard
05	American Express
06	Diners Club
07	NKL Kort
08	Multikort
09	GE Kapital
10	Gjensidige Bank
11	JCB
12	Trumf
13	Domino
14	Maestro
15	Lindexkortet

Tabell 1, Kortselskapskode pr. 20 januar 1999

function GetIssuerID : Integer;

Returverdi	tallkode for korttype ved siste transaksjon. Dersom Bankterminalkobling ikke har mottatt noen verdi fra terminalen, enten ved feil eller avbrutt transaksjon, vil –1 returneres. Denne funksjonen har normalt kun mening etter en godkjent transaksjon, men i SOFIE 5 protokollen vil denne informasjonen også være tilgjengelig ved ikke godkjente, men fullførte, transaksjoner. Dette gjelder ikke tidligere versjoner av SOFIE protokollen

Hent kontonummer (PAN)

Denne funksjonen returnerer kontonummer for kort ved siste transaksjon. Dette nummeret inneholder også kortets duplikatnummer. Duplikatnummeret er ikke en del av kontonummeret, men identifiserer kortet dersom duplikater er produsert. Dette tallet er normalt preget inn i kortet etter kontonummeret, ofte adskilt med bindestrek (-). Det er ikke alle transaksjoner som returnerer PAN, returverdien er da tom/blank.

function GetPAN : WideString;

Returverdi	inntil 19 ASCII karakterer i området H20..H7F. Dette feltet vil være blankt dersom Bankterminalkobling ikke har mottatt denne informsjonen. Denne funksjonen har normalt kun mening etter en godkjent transaksjon, men i SOFIE 5 protokollen vil denne informasjonen også være tilgjengelig ved ikke godkjente, men fullførte, transaksjoner. Dette gjelder ikke tidligere versjoner av SOFIE protokollen

Hent transaksjonstidspunkt

Denne funksjonen returnerer BBS transakaksjonstidspunkt for siste transaksjon. Denne funksjonen har kun mening etter en fullført transaksjon.

function GetTimeStamp : WideString;

Returverdi	ASCII tekst streng med format YYYYMMDDHHMMSS. Returverdien vil være blank dersom Bankterminalkobling ikke har mottatt denne informasjonen fra terminalen

***GetTimeStamp* kan kun benyttes dersom Bankterminalkobling er konfigurert til å bruke SOFIE 5 hostdata format, se konfigureringsmanual.**

Hent hostdata retur

Denne funksjonen returnerer hostdata parameteren fra kortselskapet etter at TransferAmountEx funksjonen er kjørt. Denne funksjonen har kun mening etter en fullført transaksjon.

function GetHostData : WideString;

Returverdi	inntil 40 ASCII karakterer i området H20..H7F, kan også være tom/blank. Informasjonen i denne verdien er gitt av hvert enkelt kortselskap og kan variere. Ta kontakt med Bank Axept eller aktuelt kortselskap for ytterligere informasjon

GetHostData kan kun benyttes dersom Bankterminalkobling er konfigurert til å bruke SOFIE 5 hostdata format, se konfigureringsmanual.

Denne funksjonen er ikke i bruk i SOFIE 7 men er beholdt av kompatibitetshensyn til eldre klientapplikasjoner.

Reserver beløp

Denne funksjonen brukes for å initiere reservering av beløp i et kredittkort, typisk i restaurant og hotellbransjen. Et reservert beløp må alltid gjøres opp i etterkant med oppgjørsfunksjonen, beløpet kan da justeres for eventuelt tips eller lignende. Når en reserverer et beløp gjøres dette opp mot et kundenummer (sekvensnummer). Dette kundenummeret MÅ oppgis ved endelig oppgjør. For ytterligere informasjon om reservering av beløp henvises til terminalens brukermanual.

function PreAuthorise (Amount, SeqNo : Integer; hostdata : WideString) : WordBool;

Amount	beløp som skal reservers, **beløp i øre** (1/100 NOK).
SeqNo	sekvensnummer - dersom dette er en ny kunde skal SeqNo være 0, dersom beløpet er tillegg til et allerede reservert beløp skal SeqNo være sekvensnummeret returnert etter første reservasjon. Dersom SeqNo er 0 og kunden trekker et kort med en reservasjon inne vil ternminalen i enkelte tilfeller spørre om sekvensnummer.
Hostdata	inntil 40 lesbare ASCII karakterer i området H20..H7F. Dette feltet kan også være tomt/blankt
Returverdi	True dersom dialog med terminalen er initiert. Dersom kommunikasjonsfeil, lisensfeil eller ECR allerede er i BankMode returneres False

Hent sekvensnummer

Hent sekvensnummer for siste transaksjon. Etter en reservasjon generer terminalen et sekvensnummer (kundenummer). Det kan reserveres flere beløp i et uoppgjort sekvensnummer og kjøre et felles oppgjør for alle reservasjoner. Samme kredittkort må brukes for alle tilleggsreservasjoner.

function GetSequenceNo : integer;

Returverdi	0..9999

Oppgjør

Denne funksjonen brukes for å initierer et oppgjør for et reservert beløp i et kredittkort, typisk i restaurant og hotellbransjen. Oppgjøret gjøres mot et kundenummer som er generert ved reservasjon av beløpet. Oppgjørsbeløp kan avvike fra reservasjonsbeløp for eventuelt tips eller lignende. For ytterligere informasjon om reservering av beløp henvises til terminalens brukermanual.

function Adjust (Amount, SeqNo : Integer; hostdata : WideString) : WordBool;

Amount	oppgjørsbeløp, **beløp i øre** (1/100 NOK).
SeqNo	sekvensnummer – sekvensnummer generert ved reservasjonen(e) det skal gjøres oppgjør for. Dersom SeqNo er 0 må kredittkort leses på nytt for å finne tilhørende reservasjon.
Hostdata	inntil 40 lesbare ASCII karakterer i området H20..H7F. Dette feltet kan også være tomt/blankt
Returverdi	True dersom kommunikasjon med terminalen, ellers False

Overfør kortdata

Funksjon for å overføre data for kredittkort istedet for å dra kortet i terminalens kortleser.

function TransferCardData (PAN :WideString; ExpiryDate: WideString , CVC : WideString) : WordBool;

PAN	Inntil 19 siffer kortnummer uten mellomrom
ExpiryDate	4 siffer som angir utløpsår og månde, **format må være YYMM**
CVC	Kontrollnummer, normalt trykket på kortet, normalt ikke påkrevet
Returverdi	True dersom kommunikasjon med terminalen, ellers False

Send administrasjonskode

Funksjon for å overføre administrasjonskode til terminalen. Administrasjonskoder kan blant annet brukes for å hente ut X- og Z- rapporter.

function SendCharacter (Value : unsigned char) : WordBool;

Value	HEX 0 til HEX 1F Noen forhåndsdefinert koder 16 – Avstemming 17 – Validering (OK/KLAR) 18 – Cancellation (AVBRYT) 19 – Correction (FEIL) 20 – Reversal (ANNULER) 21 – Balance Inquiry (kun for noen korttyper) 22 – X-rapport 23 – Z-rapport 24 – Send offline transactions to HOST 25 – Turnover rapport 26 – Print EOT transactions
Returverdi	True dersom kommunikasjon med terminalen, ellers False

Send karakter

Funksjon for å overføre karakterer til terminalen istedet for terminalens tastatur dersom terninalen ber om dette. Dette gjelder ikke PIN-kode, denne må tastes på terminalen.

function SendCharacter (Value : unsigned char) : WordBool;

Value	HEX 20 (blank) til HEX 7E (~)
Returverdi	True dersom kommunikasjon med terminalen, ellers False

Sett operatør id

Funksjon for å sette operatør id i terminalen, både for transaksjoner og avstemming/rapporter. Alle transaksjoner og rapporter vil ha denne operatør id inntil Bank Terminla Kobling blir restartet eller ny operatør id blir satt. Se terminalens brukermanual for ytterligere informasjon.

function SetOperID (OperID: WideString) : WordBool;

OperID	Tom streng eller 4 ASCII karakterer operatør id. Dersom formatet er feil settet operatør id til default '0000'.
Returverdi	True dersom formatet er riktig ellers False

Start avstemming

Denne funksjonen har tidligere blitt gjort manuelt fra terminalen, og terminalen har skrevet ut avtemmingsrapporten på terminalskriveren. Denne rapporten kan nå overføres til kasseapplikasjonen iform av lokalskrivermeldinger. Denne funksjonen benyttes på samme måte som ved en transaksjon, men det er ikke behov for noen kort eller pinpadkoder.

function Reconcile : WordBool;

Returverdi	True dersom ikke allerede i BankMode, ugyldig lisensnummer eller terminalen ikke er tilkoplet, ellers returneres False

Hent avstemmingsresultat

Denne funksjonen returnerer ok eller ikke ok etter at avtemmingsfunksjonen har vært kjørt. Denne funksjonen har kun mening etter at Bankterminalkobling har returnert til LocalMode

etter start avtemming funksjonen. Andre avstemmingsrapportet er også tilgjengelig ved å sende administrasjonskoder, se avsnittet Send AdminKode

function ReconcileOk : WordBool;

Returverdi	True dersom avtemming er fullført, ellers False

Avbryt

Prosedyre for å avbryte en kommando til terminalen, normalt avbryte en transaksjon eller en avstemming. Ved en transaksjon kan dette gjøres inntil kunden har tastet sin pinkode og trykket klar/godkjent, ved avstemming kan funksjonen benyttes før terminalen begynner å skrive ut avstemmingsrapporten. Denne kommandoen benyttes normalt dersom kasseoperatøren har gjort en feil ved inntasting av varer eller kunden ombestemmer seg og ikke vil betale med bankkort.

procedure Cancel;

Returverdi	ingen

Hent antall skrivermeldinger

Funksjon som returnerer antall skrivermeldinger som ikke er håndtert av systemapplikasjonen.

function GetNoOfPrinterMsgs : Integer;

Returverdi	antall uavhentede meldinger i skriver FIFO bufferet siden forrige transaksjon/avstemming ble initiert

Hent antall vindusmeldinger

Funksjon som returnerer antall lokalvindusmeldinger som ikke er håndtert av systemapplikasjonen.

function GetNoOfDisplayMsgs : Integer;

Returverdi	antall uavhentede meldinger i lokalvindu FIFO bufferet siden forrige transaksjon/avstemming ble initiert

Hent skrivermelding

Funksjon som returnerer den eldste tekst melding i skriver FIFO bufferet[1]. Disse tekstmeldingene er ikke formatert og kan inneholde koder for linjeskift, retur, kutt papir eller perforer papir. Hvordan disse koder håndteres bestemmes av systemapplikasjonen og vil være skriver avhengig.

function GetPrinterMsg : WideString;

Returverdi	en nullterminert tekst streng dersom det er uavhentede skrivermeldinger, hvis ikke returneres en tom streng

Denne funksjonen kan benyttes sammen med GetNoOfPrinterMsgs for å sjekke om det er noen uavhentede meldinger.

Hent vindusmelding

Funksjon som returnerer den eldste tekst melding i lokalvindu FIFO bufferet. Disse tekstmeldingene er ikke formatert og kan inneholde koder for linjeskift eller retur. Hvordan disse koder håndteres bestemmes av system applikasjonen og vil være lokalvindu avhengig.

[1] se avsnitt FIFO tekstbuffer

function GetDisplayMsg : WideString;

Returverdi	en nullterminert tekst streng dersom det er uavhentede vindusmeldinger, hvis ikke returneres en tom streng

Denne funksjonen kan benyttes sammen med GetNoOfPrinterMsgs for å sjekke om det er noen uavhentede meldinger.

Sett skriverstatus

Funksjon for å sette status for den lokale skriveren. Denne funksjonen må benyttes dersom skrivermeldinger ikke skal komme ut på terminalens skriver.

procedure SetPrinterStatus (status : OLEPrinterStatusEnum);

OLEPrinterStatusEnum har tre alternative verdier[2]:

PrinterOk	tallverdi 0, lokalskriver er klar, meldinger vil ikke skrives på terminalskriveren
PrinterBusy	tallverdi 1, lokalskriver er opptatt, ITU vil prøve en gang til etter en kort stund.
PrinterFailure	tallverdi 2, lokalskriver feil, meldingen vil bli skrevet på terminalskriveren. Normalt vil hele kvitteringen bli skrevet ut på terminalens skriver selv om ECR rapporterer PrinterFailure kun på en av skrivermeldingene.
Returverdi	ingen

Sett lokalvindustatus

Funksjon for å sette status for den lokale visningsenheten.

procedure SetDisplayStatus (status : OLEDisplayStatusEnum);

OLEDisplayStatusEnum har to alternative verdier[3]:

DisplayOk	tallverdi 0, lokalvindu er klar
DisplayBusy	tallverdi 1, lokalvindu er opptatt, ITU vil prøve en gang til etter en kort stund.
Returverdi	ingen

Merk at noen terminaler ikke får utført en transaksjon dersom lokalvinduet har status DisplayBusy.

Skjul brukerpanel

Funksjon for å skjule Bankterminalkobling's brukerpanel. Brukerpanelet vil normalt ikke være ønskelig å vise på skjermen og systemapplikasjonen kan bruke denne funksjonen ved oppstart av systemet. Dersom denne funksjonen kjøres vil ikke applikasjonen få et ikon på oppgavelinjen heller, brukeren av maskinen vil ikke ha kjennskap til at denne applikasjonen kjører. **Bankterminalkobling er default usynlig når den startes embedded/automated.**

[2] Hvordan terminalen behandler opptatt og feil status kan være terminal avhengig og må sjekkes mot leverandøren av terminalen. Oppførsel som beskrevet her er i henhold til SOFIE protokollens det refereres til under avsnitt 0

[3] som for SetPrinterStatus

procedure Hide;

Returverdi	ingen

Vis brukerpanel

Motsatt funksjon av Hide[4].

function Show;

Returverdi	ingen

Les/Skriv parameterverder

Funksjon for aksess til interne verdier i BTKProgram, typisk konfigurasjonsparametre. Ved å aksessere konfigurasjonsparametere direkte overstyrer en konfigurasjonen i Registry. Alle parameter- navn og -verdier er som tekststrenger, de må dekodes av klientapplikasjonen dersom verdien ikke er en tekststreng. For at parameterendringer skal aktiveres må **'Modified' settes til '0'**, dette tilsvarer 'Bruk' knappen men endrede verdier blir ikke skrevet til Registry, se også avsnitt Parametere med spesiell betydning.

Parameternavn er ikke case-avhengige.

Function GetProperty(PropName : WideChar) : WideChar;

Returverdi	Parameterverdi som streng. Dersom PropName ikke finnes returneres '<NOPROP>'

function SetProperty(PropName : WideChar; Value : WideChar) : WordBool;

Returverdi	True dersom ok, ellers false

For å få en liste over tilgjengelige paramtere kan en bruke GetProperty med **'properties'** som PropName. Funksjonen returnerer da en semikolon separert liste over alle parametre.

[4] se avsnitt for funksjonen Hide under 0

Parametere med spesiell betydning

PropName	Verdi	Beskrivelse
Properties	Semikolonseparert streng	Liste over alle supporterte PropNavn, skilt med semikolon ';'. Ikke alle parametere er skrivbare.
Modified	'0','1'	Etter enhver endring av en konfigurasjonsparameter settes 'Modified' til '1'. For at endringer skal aktiveres må 'Modified' settes til '0'. Dette tilsvarer 'Bruk' knappen bortsett fra at endringene blir ikke lagret i Registry ('Bruk' knappen lagrer alltid innholdet i feltene i konfigurasjonsvinduet). Å sette 'Modified' til andre verdier enn '0' har ingen innvirkning.
SystemType	Streng, inntil 13 karakterer	Samme som SystemIdentifikasjon. Alle felter i Transaksjonsidentifikasjon sendes til BBS og gir sporbarhet i BBS's logger.
BTKLicense	Lisensstreng generert av TORDIVEL AS	Ved å sette BTKLicense kan en overstyre standard lisensiering (hvor terminalId er knyttet opp mot et unikt lisensnr). En BTKLicense universallisens genereres ut fra SystemType. Det må være samsvar mellom SystemType og BTKLicense, den er case-avhangig, ellers vil ikke lisensen virke. Det anbefales at klientapplikasjonen setter SystemType før BTKLicense.

Funksjonell bruk

I de fleste tilfeller er det kun initiering av transaksjoner som systemapplikasjonene har behov for, for deretter å vente med all annen aktivitet inntil transaksjoner er utført, uavhengig av om transaksjonen er godkjent av BBS eller ikke. Dersom transaksjonen ikke er godkjent må operatøren av kasseapparatet enten initiere en ny transaksjon dersom kunden ønsker å prøve et annet kort eller kunden tar kontant oppgjør eller avvises. Tilsvarende ved avstemming, det er normalt ingen andre operasjoner som foregår mens avstemmingsrapporten skrives ut.

En enkel rutine mot serveren skulle være lett å implementere i de fleste utviklingsverktøy. Selve Bankterminalkoblingen er *hendelsesstyrt*, den vil ikke låse noen Windows ressurser slik at det er opp til systemapplikasjonen å avgi prosessorkraft til eventuelle andre applikasjoner som skal kjøre samtidig. Dette er vanlig i de fleste moderne hendelsesstyrte applikasjoner som kjører under operativsystem som er 'multi tasket', slik som Windows 95/98 og Windows NT/2000/XP. Selve hånteringen av protokollen er ikke særlig ressurskrevende da kommunikasjonsraten er lav og meldinger kommer i en slik sammenheng sakte. Men man må likevel merke seg at Windows ikke er et fullt ut 'multi tasket' 'time sliced' operativsystem slik at løkker som kjører normalt vil kjøre ferdig når de først får prosessor aksess. Med dette tatt i betrakning bør ikke serveren kjøre på samme maskin som slike prosessorkrevende applikasjoner, noe som i de fleste tilfeller heller ikke er tilfellet.

I SDK installasjonen følger det med 2 eksempel for bruk av OLE Automation grensesnittet, ett i Visual Basic og ett i Visual C++. I Visual Basic eksempelet benyttes en løkke for å vente på at Bankterminalkobling skal returnere til Local Mode, men for å ikke 'låse' denne og eventuelt andre applikasjoner gjøres det i hver omgang i løkken ett kall til **DoEvents**. Dette kallet gjør blant annet at alle vinduer i kasseapplikasjonen oppdateres. Tilsvarende kan gjøres i Delphi ved kall til applikasjonsobjektets metode **Tapplication.ProcessMessages**. I Visual C++ eksempelet benyttes imidlertid en bedre metode da det ved start av en transaksjon settes opp en timerfunksjon som sjekker om Bankterminalkobling har returnert til Local Mode. Denne fremgangsmåten belaster operativsystemet mye mindre enn i en løkke.

Pseudokode for transaksjon

Her følger et pseudokodeeksempel for en transaksjon med varekjøp for 999.50 kroner og kontantuttak 500 kroner. Kodeeksemplet tar i bruk alle funksjoner som er nødvendig for å utføre en transaksjon.

BAX – referanse til BankAxeptSrv

```
IF BAX.Connected AND BAX.LicenseVerified AND NOT BAX.BankMode THEN
  BAX.SetPrinterStatus status
  BAX.SetDisplayStatus status
  IF TransferAmount(99950, 50000) THEN
    WHILE BAX.BankMode DO
      IF BAX.GetNoOfPrinterMsgs > 0 THEN
        print BAX.GetPrinterMsg
        BAX.SetPrinterStatus status
      END
      IF BAX.GetNoOfDisplayMsgs > 0 THEN
        display BAX.GetDisplayMsg
        BAX.SetDisplayStatus status
      END
      yield_other_applications
    END
    IF NOT BAX.TransactonOk THEN
      handle_transaction_rejected
    END
  END
END
```

Funksjonene print og display er brukerdefinerte, men vil normalt skrive ut meldingen på henholdsvis en lokaltavlesningsenhet og kassekvitteringen. Brukerapplikasjonen må også oppdatere skriver- og lokalvindustatus slik at Bankterminalkobling returnerer aktuell skriver- og lokalvindustatus når slike meldinger skal håndteres.

Denne fremgangsmåten kan egne seg godt i Visual Basic og Delphi, men et bedre alternativ enn å kjøre i en løkke, er å sette opp en timerrutine som sjekker om transaksjonen er ferdig. I denne rutinen tas også eventuelle meldinger hånd om, samme kode som kjøres inne i løkka. Dette er mere i tråd med moderne hendelsesstyrt design.

Begge disse metodene er vist i eksemplene som følger med installasjonen av Bankterminalkobling, løkkefunksjonalitet i Visual Basic, timerfunksjonalitet i Visual C++.

Skriptmodul

Fra versjon 2.1 er gratisproduktet Python integrert i BTKProgram. Python er et fleksibelt og kraftig scriptspråk som gjør at BTKProgram kan utvides og tilpasses spesielle krav. BTKProgram inneholder 2 interne Python moduler som gir brukeren aksess til BTKPrograms interne objekter. For mer informasjon om Python, se http://www.python.org. For å integrere Python skripting i BTKProgram må **Python 2.3** eller høyere være **installert**. Dersom Python er installert på maskinen vil BTKProgram vise en ekstra flik ved siden av konfigurasjonsfliken.

Skriptmodell

Python er integrert i BTKProgram i en Event-modell, ved spesielle hendelser/eventer i BTKProgram kalles navngitte skript dersom de er definert og aktivert. Hva skriptene utfører er gitt av skriptets innhold. Selve skriptene er definert i en ekstern tekstfil som lastes ved

oppstart av BTKProgram. Ved oppstart kan en kjøre en del initiell kode. Den initielle koden kan definere variable i Pythons navnerom som vil være globale for alle skript.

OnInit

Signatur : def init():

Denne eventen kalles ved oppstart etter at BTKProgram er startet men etter at eventuell initiell kode ved lasting av skriptfilen er kjørt. Denne eventen blir også kalt dersom man trykker 'Last' knappen eller endrer skriptfil.

OnClose

Signatur : def close():

Denne eventen kalles før avslutning av BTKProgram. Denne eventen blir også kalt dersom man trykker 'Last' knappen eller endrer skriptfil.

OnPrinterMsg

Signatur : def printerMsg(msg):
msg = string

Denne eventen blir kalt for hver melding som går til skriver. Man kan f.eks. enkelt sende meldingen til annet eksternt utstyr. Msg kan inneholde karakterer med spesiell betydning, f.eks karakterer som må oversettes til norsk eller kommandoer for papirkutt/perforering, dette er det likt som funksjonen GetPrinterMsg().

OnDisplayMsg

Signatur : def displayMsg(msg):
msg = string

Tilsvarende som OnPrinterMsg men for displaymeldinger.

OnLocalMode

Signatur : def localMode():

Denne eventen blir kalt hver gang terminalen går bra tilstanden BankMode til LocalMode. Brukeren (skriptene) må selv holde rede på om tilstandsendringen kommer på bakgrunn av en transaksjon, avstemming eller en annen administrativ/teknisk operasjon i terminalen, se avsnittet Local Mode / Bank Mode.

OnTimer

Signatur : def timer(name):
name = string

BTKProgram har en intern pythonmodul som gir brukeren mulighet til å definere timere. Når en brukerdefinert timer 'trigger' kalles OnTimer eventen. Brukeren kan da bruke 'name' til å identifisere hvilken timer som har 'trigget' og eventuelt aksessere timerens attributter.

Eks:
```
def OnTimer(name):
  import stdtdv
  t=stdtdv.Timer(name)
  t.active=False    #oneshot timer
  if t.name=='POLLER':
    #do some polling stuff…
```

Interne Pythonmoduler

stdtdv – standard tordivel modul

stdtdv inneholder timere som brukeren kan benytte til periodiske aksjoner i BTKProgram.

Timer objektet

Constructor : stdtdv.Timer(name,[interval=1000,active=1)

Timer objekter inneholder referansetelling, dersom en prøver å opprette en timer som allerede finnes (name er identisk med en allerede opprettet instans) får en fra constructoren et python object med referanse tll den eksisterende timeren. I slike tilfeller ignoreres parametene interval og active dersom de er gitt.

Attributer (properties)

name – string, readonly, definert i constructor
active – bool read/write
interval – int – read/write – oppløsning I millisekunder
tag – int – read/write - brukderdefinert attributt

Ingen metoder

BTK modulen

BTK – intern modul som inneholder objektet BTK som gir aksess til betalingsfunksjoner.

Constructor : BTK.BTK()

Attributter (properties) – ingen

Metoder
BTK objektet har eksakt samme grensesnitt som COM grensesnittet, alle COM metoder er også implementert i Python. Merk at alle metoder er case sensitiv.

Eks, start en transaksjon:

```
import BTK
ongoingTransaction=False        #internal userdefined application state
btk=BTK.BTK()
if btk.Connected() and not btk.BankMode():
  if btk.TransferAmount(15000,50000): #purchase=150 kr, cashback=500 kr
    ongoingTransaction=True
```

```
#-----------------------------------------------
#Autogenerated script for BTKProgram
#Created 18.08.2005 12:17:12

#-----------------------------------------------
# OnInit
def init():
  pass

#-----------------------------------------------
# OnClose
def close():
  t.active=False
  client.close()

#-----------------------------------------------
# OnPrinterMsg
def printerMsg(msg):
  pass

#-----------------------------------------------
# OnDisplayMsg
def displayMsg(msg):
  pass

#-----------------------------------------------
# OnLocalMode
def localMode():
  global processing
  t.active=True
  if processing:
    if btk.TransactionOk():
      print 'Transaction OK Purchase %.2f Cahsback %.2f' % (purchase,cashback)
    else:
      print 'Transaction REJECTED Purchase %.2f Cahsback %.2f' % (purchase,cashback)
  while btk.GetNoOfPrinterMsgs()>0:
    print btk.GetPrinterMsg()
  processing=False

#-----------------------------------------------
# OnTimer
def timer(name):
  global processing
  global purchase
  global cashback
  ok,(purchase,cashback)=client.poll()
  if ok:
    print 'Transaction request Purchase %.2f Cashback %.2f' % (purchase/100,cashback/100)
    if btk.Connected() and not btk.BankMode():
      if btk.TransferAmount(purchase,cashback):
        processing=True
    else:
      print 'Terminal BUSY or not connected'

#module initialization
print 'BTKProgram Python Initialization'

import sys
sys.path.append('\\scripts') #append path to find btksock.py

import stdtdv,BTK,btksock

processing=False  #flag for ongoing transaction
purchase=0        #purchase amount
cashback=0        #cashback amount

client=btksock.ECRClient() #create a ECRClient object for receiving transaction messages

btk=BTK.BTK()           #create the btk object
btk.SetPrinterStatus(0) #signal printer ok to avoid copy on terminal printer
btk.SetDisplayStatus(0) #to signal display ok

t=stdtdv.Timer('POLL',500,1) #create a timer with polling period 500 ms
```