Meddelandesystem för mobilt internet

Patrik Wållberg (c96pwg@cs.umu.se)

Handledare:
Jerry Eriksson

Abstract

Mobile internet is rapidly becoming an important alternative to traditional web access methods. One of the most important aspects of mobile internet in the financial sector is the possibility of alerting a user when certain conditions are met. A prototype messaging system was created to service a wide range of messaging needs brought on primarily by the mobile internet evolution. This messaging system combined with a WAP client, supports a new alert-driven trading experience where a user performs the actual trading by reacting to trading tips and price alerts. This way of trading can attract new users who prefer to receive information when it is available rather than find it by themselves. The user adds, removes and modifies subscriptions (i.e. alerts) through a web client or a WAP client. The resulting messaging system is a modular solution implemented in the Java programming language and using an Oracle database. The clients were implemented using JSP, HTML and WML. SMS messaging functionality was implemented using Nokia's CIMD2 API. After this project was finished, the complete system was successfully demonstrated at the SIA Exhibition in New York.

Innehållsförteckning

1. Inledning

Uppgiften i detta examensarbete var att utveckla ett meddelandesystem för mobilt internet åt företaget Interbizz Financial Systems. Arbetet utfördes av mig, Patrik Wållberg, student vid datavetenskapliga institutionen, samt två kollegor på företaget, Alexander Sehlin och Peter Sönnergren. Största delen av detta arbete utfördes år 2001.

2. Bakgrund

I drygt 120 år användes telefonen för att tala i. Men i slutet av 80-talet hände något, då blev telefonen bärbar. Mobiltelefonen fick flera nya uppgifter och är nu ett verktyg som erbjuder användaren information i många olika former. Den blir en plånbok, kommunikationscentral och fjärrkontroll där användaren med smarta tjänster kan styra information, nyheter och erbjudanden etc. Allt detta blir möjligt nu när tele- och datakommunikation har växt samman i något som brukar kallas mobilt Internet.

Den globala finansmarknaden och dess aktörer går precis som så mycket annat idag mot ett allt större användande av mobilt Internet. Trådlösa tjänster för mobilt Internet utvecklas och erbjuds nu i ett accelererande tempo. Idag finns det inte många människor i Sverige mellan 15-60 år som inte har en mobiltelefon som stöder SMS meddelanden. GSM systemets efterföljare GRPS och 3G ökar ytterligare intresset och bör leda till ännu flera användningsområden för mobila internettjänster.

För att företag inom finansbranschen ska hänga med i denna utveckling mot mobilt Internet måste de nå ut till de mobila aktörerna på marknaden. Förutom mobila WAP klienter så behövs bland annat meddelandesystem som kan förse aktörerna med viktig information i realtid. Ett sådant meddelandesystem skapar tillsammans med mobila klienter ett nytt sätt att agera på marknaden. Mobila aktörer kan då istället för att själva leta reda på information de behöver, låta meddelandesystemet sända dem önskad information direkt när informationen finns tillgänglig.

Företaget bakom uppgiften till detta examensarbete, Interbizz Financial Systems, bygger finansiella system för värdepappershantering över Internet riktat mot en global finansmarknad. Företagets system används bland annat av de tre största världsbankerna samt Englands och Sveriges största Internetmäklare. År 2001 hade företaget omkring 100 anställda.

2.1 Meddelandesystem

Ett meddelandesystem är ett system som automatiskt tar emot, hanterar och skickar vidare meddelanden med någon form utav information. Meddelandena kan baseras på information genererad från olika typer av källor och ha specifika mottagare. System av denna typ används idag i allt större grad av både företag och kunder i många branscher.

Ett system som klarar av att ta emot meddelanden från olika typer av källor, komplettera dessa, och sända dem vidare till specifika mottagare i realtid blir mycket användbart i många situationer. Ännu mera användbart blir systemet om det hanterar både mobila och stationära källor och mottagare.

2.2 Disposition av slutuppsats

Denna rapport inleds med en problembeskrivning i avsnitt 3. Problembeskrivningen innehåller en beskrivning av uppgiften, och de mål som satts upp presenteras. Därefter följer i avsnitt 4 en presentation av de metoder som använts under examensarbetet och de olika val som gjorts. I avsnitt 5 presenteras kort historik kring GSM samt information om SMS (Short Message Service) och det SMS gränssnitt som använts. Det gränssnitt som använts heter CIMD2 och ägs av Nokia. Jag har jobbat allra mest med SMS delen i det meddelandesystem som skapats i detta examensarbete och har därför fördjupat mig i denna teknik. Fördjupningsdelen i denna rapport belyser detta område. En beskrivning av genomförandet av detta examensarbete hittas i avsnitt 6. I avsnitt 7 presenteras alla resultat och i följande avsnitt 8 återfinns en avslutande diskussion kring resultatet.

3. Problembeskrivning

Målet med examensjobbet är att utveckla och implementera en fungerande prototyp för ett komplett meddelandesystem där modularitet och flexibilitet eftersträvas. Systemet ska designas så att det enkelt kan byggas ut med nya moduler som hanterar nya in- och utkanaler samt olika typer av databaser. Alla faser av projektet ska dokumenteras ordentligt.

Meddelandesystemet ska ha en inkanal som tar emot information från finans-marknaden via Interbizz finanssystem ROX (Realtime Order eXecution). Primärt är det information om kursförändringar på värdepapper som kommer att tas emot i realtid från en simulerad marknad. Systemet ska direkt kontrollera all mottagen information mot en databas. Om det visar sig finnas mottagare som anmält intresse för aktuell information ska ett meddelande genereras och sändas till mottagarna.

Systemet ska kunna kommunicera med både stationära och mobila mottagare. Den utkanal som systemet ska använda sig av för att nå mottagarna ska vara en SMS-kanal. En Webb-klient ska även skapas där mottagarna, det vill säga användarna, ska kunna logga in och anmäla intresse för olika händelser och förändringar på specifika

värdepapper. För statuskontroll av systemet ska ett administrationsverktyg skapas. Verktyget ska ha ett grafiskt gränssnitt och det ska kunna kopplas till ett körande meddelandesystem från vilken Internetansluten dator som helst.

Nedan följer en beskrivning i punktform över mina huvudsakliga uppgifter i detta projekt. Värdet som anges i procent är mitt personliga ansvar för aktuellt delmoment.

- Utredning, inläsning och förstudier för projektet. Exemplevis XML, JSP, SMS, JDBC. (50%)
- Kravspecifikation. (50%)
- Analys och design. (33%)
- Implementation. (33%) Varav SMS-modul och Webb-klient (100%)
- Testning. (33%)
- Dokumentering för projektet. (33%)
- Dokumentering för examensarbetet. (100%)

3.1 Avgränsningar

Meddelandesystemet som skapas ska kunna klara av många olika in- och utkanaler. Som det nämndes i problembeskrivningen så avgränsar vi uppgiften till endast en inkanal samt en utkanal. Utkanalen som ska vara en SMS kanal avgränsas till att klara av endast ett valfritt SMS protokoll.

Ytterligare avgränsningar är att vi inte har några höga prestanda- och säkerhetskrav på systemets olika delar i denna första version som examensjobbet omfattar. I denna första version är det viktigare att systemet fungerar i demonstrationssyfte än att exempelvis implementera stark kryptering för SMS kanalen.

4. Metodbeskrivning

De metoder som använts under prototyputvecklingen redovisas kort i detta avsnitt. Till metoderna inkluderas exempelvis programmeringsparadigmer samt modeller och notationer som används vid programvaruutveckling.

4.1 Java

Den programmering som utförts är gjord i programspråket Java. Java valdes för att det är ett relativt enkelt men kraftfullt objektorienterat språk som bidrar till hög produktivitet. En annan viktig orsak var att alla inblandade i detta projekt hade mera kunskaper i Java än i exempelvis C++ eller något annat objektorienterat språk. Att Java är plattformsoberoende var också en viktig anledning [20,21].

4.2 Objektorientering

Programmering, analys och design av alla delar i systemet har utförts objekt-orienterat. Detta var möjligt eftersom det objektorienterade programmerings språket Java användes för att utveckla systemet. Den objektorienterade paradigmen innehållande koncept som arv, inkapsling, polymorfism och dynamisk bindning har som syfte att ge flexibla och modulära system, samt hög grad av återanvändbarhet och produktivitet. Paradigmen stödjer hela livscykeln för ett mjukvarusystem och kan användas i alla faser. Terminologin är densamma i alla de olika faserna för att underlätta kommunikation och ger ökad spårbarhet [1].

4.3 ER-modellen

Entity-relationship-modellen (ER-modellen) är en populär datamodell för modellering på hög nivå. Modellen och varianter av den används frekvent vid begreppsmässig design av databasapplikationer, och många designverktyg för databaser använder dess ingående begrepp. Entity-relationship modellering (ER-modellering) är ingen ny teknik. Den har funnits och använts sedan 1976 [1,2]. Vid databasmodellering av databaser till det utvecklade meddelandesystemet har ER-diagram använts. Dessa har sedan konverterats till motsvarande relationsschema innan realisering.

4.4 Treskiktssystem

Vid utveckling av databasapplikationer som ska vara åtkomliga från webben kan en treskiktad modell användas. Flerskiktade databasapplikationer består vanligtvis av:

- En klientsida med användargränssnitt skrivet i Java och/eller HTML.
- En eller flera komponenter för mellanskiktet vilka håller affärslogik och vyer till affärsobjekt.
- Tabeller på databasservern som lagrar underliggande data.

Syftet med treskiktsmodellen är att dela upp funktionaliteten i flera lager. Det översta lagret, det grafiska gränssnittet bestående av web-sidor, kan på så vis enkelt exempelvis läggas ut på kontrakt. För att kunna utföra lite mer avancerade funktioner innehållandes affärslogik och access mot databaser etc. skrivs sådana delar som exempelvis Java-komponenter. Det ger bland annat bättre prestanda och säkerhet samt förenklar komplexiteten i de web-sidor som anropar Java-komponenterna. De delar som ingår i den treskiktsmodell som använts i detta examensarbete beskrivs kort nedan.

4.4.1 JSP/HTML

Vid utveckling av grafiska gränssnitt som ska vara åtkomliga via Internet kan HTML plus något programmerings- eller skript-språk användas. Valet av språk har exempelvis att göra med vilken serverprogramvara som erbjuds. Eftersom vi valt att jobba med programmeringsspråket Java i övrigt föll det sig naturligt att välja Java Server Pages (JSP) för vår del. Vi hade inget krav på val av serverprogramvara. JSP är en teknik framtagen av Sun Microsystems där javakod och javascript kan blandas med html kod för att göra sidorna dynamiska. När ett anrop sker till en JSP-sida exekverar Web-servern sidan som tolkas och skickas över som vanlig HTML [3].

Detta betyder att HTML används för att bygga statiska web-sidor medan JSP kan användas för dynamiska sidor och lite mer avancerade uppgifter. Det skriptspråk som används i JSP är Suns egna JavaScript. Allmänt så kan implementation av logik i exemelvis skript delas upp i de som körs på klienten och de som körs på en server. En fördel med att använda skript som körs på server-sidan är att det ger snabbare och stabilare sidor som är kompatibla i alla web-läsare. Ofta behöver dock inget val göras, det beror helt på skriptets uppgifter. För databasaccess måste skriptet, logiken, ligga på en server och för grafiska tillämpningar som att följa muspekaren måste skriptet ligga hos klienten. Men vid t.ex. en kontroll av att alla fält är rätt ifyllda finns det en valmöjlighet. Dock väljs allt som oftast server-skript eftersom användaren ibland av säkerhetsskäl kan stänga av klient-skript.

4.4.2 Java/JDBC

Komponenterna i mellanskiktet implementerades i programmeringsspråket Java av bland annat tidigare nämda anledningar. Java innehåller för övrigt ett relativt lättanvänt API för att jobba mot databaser, JDBC (Java DataBase Connectivity). API är förkortningen för Application Programming Interface, en uppsättning metoder eller funktioner som ett program kan anropa för att utföra olika saker. De delar i meddelandesystemet som kommunicerar med databasen implementerades med hjälp utav JDBC.

JDBC är alltså ett programmeringsgränssnitt (API) för att kommunicera med databaser med hjälp av Java. Förutom själva gränssnittet består JDBC även av en så kallad drivrutin (driver). Med denna tvådelade arkitektur uppnås ett extra lager av abstraktion, vilket innebär att utformningen av javaprogrammet blir densamma oavsett vilken databas som anropas. Ett byte av databas kräver endast ett byte till motsvarande drivrutin. Eftersom många av databasleverantörerna erbjuder en JDBC-drivrutin till sina databasservar kan denna lösning betraktas som både produkt- och plattformsoberoende [4].

4.4.3 SQL/Oracle RDBMS

Structured Query Language (SQL) är en standard för att skapa, radera och modifiera relationsdatabaser. Det är ett väldigt kraftfullt språk som bygger på att användaren ställer frågor som beskriver det data användaren vill se. Det finns en rad system som bygger på SQL. Några av dessa är: Access, DB2, Informix, MySQL, Oracle, SQL Server, Sybase, etc. Vissa av dessa har implementerat egna dialekter av SQL, men de vanligaste anropen ser likadana ut på de flesta system. Med de vanligaste SQL-kommandona Select, Insert, Delete, Create kan det mesta göras med en databas.

Förstahandsvalet av databasserver föll på Oracle tack vare dess stabilitet och prestanda samt stora spridning inom finansbranschen och tilltänkta intressenter för meddelandesystemet. Oraclelicenser är dock väldigt dyra men det var inget hinder för Interbizz. Oracle är den marknadsledande databasservern från världens näst största programvaruföretag med samma namn. Från och med Oracle8 är den en objekt- och relationsdatabas och Oracle8i är dessutom speciellt designad för internetutveckling. Oracle är snabb, skalbar och driftsäker [5].

4.5 XML

Som internt protokoll i meddelandesystemet används XML. XML valdes för att det är lätt att arbeta med och är lättläst. XML gör också mycket enkelt att kommunicera med andra system över exempelvis Internet. XML är en förkortning som står för "Extensible Markup Language". XML är ett standariserat språk som beskriver en syntax för hur man kan skapa egna märkordsuppsättningar. Dessa märkords-uppsättningar kan sedan användas för att beskriva och strukturera information. XML är ett metaspråk. Man kan alltså ta fram sitt "eget HTML" som är särskilt anpassat efter den slags information man vill presentera. XML utvecklades i en arbetsgrupp som bildades 1996 under överinseende av W3C (World Wide Web Consortium). XML 1.0 specifikationen blev officiell rekommendation i februari 1998.

Den organisation som står bakom XML specifikationen är W3C, en viktig aktör med fler än 250 medlemsorganisationer [6,19]. WC3 är inte någon standardorganisation, utan officellt är XML specifikationen en rekommendation, men den kan dock i praktiken jämställas med en standard. XML är nära besläktad och helt kompatibel med ISO-standarden SGML (Standard Generalized Markup Language). Ett syfte med XML är att det ska vara ett globalt och universellt format för att beskriva data och information på ett sätt som är oberoende av enskilda applikationer, operativ-system och leverantörer. Man kan skapa, spara och läsa XML dokument på olika operativsystem och med olika programvaror från olika tillverkare. Detta borgar för en bättre framtidssäkring.

5. SMS

SMS står för Short Message Service och är en teknik som ger mobiltelefonanvändare i GSM-system möjlighet att skicka och ta emot textmeddelanden. Tekniken för att skicka SMS har funnits sedan 1992. Utan egentlig marknadsföring och när det mest pratades om tekniker som WAP, GPRS, och 3G, fick SMS sitt genombrott. I april 1999 skickades en miljard SMS i Europa och under december 2000 skickades nästan nio miljarder SMS. Användningen har ökat sedan dess och spås fortsätta till åtminstone slutet av år 2004 [7].

5.1 GSM historia

GSM-standardiseringen startades då CEPT (Conference of European Posts and Telecommunications) bildade den grupp, Groupe Spécial Mobile (GSM), som skulle ta fram det nya mobila telefonsystemet. GSM höll sitt första möte i Stockholm december 1982. När ETSI (European Telecommunications Standards Institute) bildades 1988 ombildades GSM till SMG (Special Mobile Group). Ett första mål var att slutföra alla specifikationer så att nätverken kunde börja byggas under 1991.

De första specifikationerna för GSM 1800 MHz (Phase 1) som började utkristalliseras redan 1988 innehöll alla de viktigaste tjänsterna såsom: telefonitjänster, datatransport, faxtjänster och SMS-tjänster där både SMS-PP [8] och SMS-CB [9] ingick. Läs mer om dessa under rubriken 5.3. År 1997 fanns det nya digitala systemet i drift i samtliga världsdelar. Systemet kallades GSM, förkortningen står idag för Global System for Mobil Communication.

5.2 Funktionalitet

Enligt GSM-standarden [10] uppdelas frekvensutrymmet i 124 radiokanaler. Varje radiokanal är därefter uppdelad i åtta tidluckor. Sju av dessa tidluckor används som trafikkanaler. Den åttonde tidluckan används för all form av signalering som behövs för att etablera och upprätthålla förbindelser på de övriga sju trafikkanalerna. SMS skickas enligt GSM-specifikationen på trafikkanalen. Ett SMS-paket på den kanalen är uppbyggt av 140 oktetter, med 7-bitars teckenkodning ger detta 160 tecken (140 * 8 / 7). Det första SMS-meddelandet skickades i slutet av 1991 från en PC till en mobiltelefon efter att SMS introducerats i GSM-specifikationen .

GSM-standardens uppbyggnad gör att endast en begränsad del av den totala radiokapaciteten kan användas för SMS. Därav den begränsade meddelande-storleken. Fördelen blir att SMS-meddelanden skickas parallellt övrig data vilket betyder att SMS kan mottagas under pågående samtal. Meddelanden överförs med mellanlagringsteknik, dvs. meddelandet överförs från sändaren till en meddelande-central (SMSC). Denna SMSC (Short Message Service Centre) gör sedan upprepade försök att leverera meddelandet till mottagaren. Detta handhavande brukar även kallas ”store-and-forward”. Ett meddelande där inte omgående leverans kan ske lagras hos SMSC normalt i 72 timmar. Användaren kan dessutom få ett kvitto när meddelandet levererades och behöver inte fundera på omsändning etc.

5.3 Användning

När man talar om SMS menar man ofta SMS-PP (SMS Point-to-Point) [8]. Detta lämpar sig mycket väl då man vill ha kommunikation med geografiskt utspridda GSM-telefoner, eller då meddelandena sänds fördelat i tiden. Exempel på sådana tjänster kan vara transportledning, vaktbolag, hemtjänst, larm till driftpersonal samt styrning och övervakning av teknisk utrustning etc. I tillämpningar där stora mängder SMS ska levereras samtidigt till GSM-telefoner som befinner sig inom ett begränsat geografisk område passar SMS-PP sämre. För det ändamålet finns en annan tjänst i GSM-specifikationen, SMS Cell Broadcast (SMS-CB) [9].

5.4 SMS-utskick från externa applikationer

Med extern applikation menas en programvara som skickar data till en SMSC vilken i sin tur har kontakt med GSM PLMN (GSM Public Land Mobile Network). SMSC fungerar som en lagringscentral mellan applikation och radionät. En SMSC kan vara kopplad till flera PLMN, ett för respektive operatör. Inom varje PLMN kan det dessutom finnas kopplingar till flera MSC:er (Mobile Service switching Centre). MSC är en vanligt förekommande enhet i GSM-system med ytterst viktiga uppgifter.

Ibland brukar funktioner för SMS även integreras i en MSC eller SMSC. Det är framförallt de två enheterna SMS-GMSC (Short Message Service Gateway Mobile services Switching Centre) och SMS-IWMSC (Short Message Service Interworking MSC) som brukar nämnas. Skillnaden mellan de två är att SMS-GMSC används när ett SMS överförs från SMSC till en MS (Mobile Station) och SMS-IWMSC när SMS överförs från MS till SMSC. Nedan beskrivs hur ett SMS skickas av en applikation.

Applikationen kommunicerar med en SMSC enligt dennes API.

- SMSC skickar vidare meddelandet till SMS-GMSC.
- SMS-GMSC frågar HLR (Home Location Register) efter nödvändig routinginformation till MS.
- SMS-GMSC skickar meddelandet till rätt MSC.
- MSC frågar VLR (Visitor Location Register) efter nödvändig routinginformation.
- MSC skickar meddelandet till rätt BSC (Base Station Controller).
- Denna BSC skickar till sina BTS:er (Base Transceiver Station).
- MS svarar om den är påslagen och har täckning.
- BSC:en fortsätter att kommunicera med den BTS där svaret kom från.
- BTS:en skickar meddelandet till MS:en.

5.5 SMS i framtiden

En intressant fråga är hur framtiden ser ut för användningen av SMS tjänster. Detta speciellt med tanke på att GSM nätet utvecklats och numera allt oftare används för tjänster som utnyttjar tekniken GSM Packet Radio Service (GPRS) [11]. GPRS möjliggör förmedling av datapaket med hög hastighet och tjänster som utnyttjar denna teknik kommer förmodligen att konkurrera ut SMS tjänster framöver. En annan sak som kommer att bidra till att SMS tjänsterna slutligen bör gå i graven är de nya 3G näten med dess tillhörande tjänster och tillämpningar.

SMS tekniken har några unika egenskaper som att ett meddelande sparas i meddelande centralen om inte mottagaren går att nå, en bekräftelse på om ett skickat meddelande kommit fram kan fås samt att samtidig överföring tillsammans med tal, data och fax tjänster är möjlig. Dessa egenskaper finns inte i GPRS tekniken. En stor begränsning med SMS meddelanden är att de inte kan innehålla mer än 160 tecken.

Enligt GSMWorld [12] kommer SMS tjänster att användas flitigt åtminstone till och med år 2005. Detta eftersom mobiltelefoner, infrastruktur, specifikationer och kundernas intresse för tjänsterna finns och fungerar. Allt eftersom kommer dock användare att koppla upp sig mot nätverk som erbjuder mera avancerade datatjänster och skaffa mobiltelefoner som stöder dessa tjänster. Populära SMS tjänster har dock möjlighet leva vidare parallellt med de nya tjänsterna eftersom de nya telefonerna och protokollen såsom Wireless Application Protocol (WAP) kommer att vara kompatibla med flera olika standarder och tekniker för överföring av data.

5.6 MMS

En utvecklad variant av SMS är Multimedia Messaging Service (MMS). MMS är som namnet antyder en teknik som gör det möjligt att skicka och ta emot meddelanden bestående av text, ljud, bild och video. Det är organisationerna WAP-Forum och 3GPP som tillsammans tagit fram denna standard. MMS använder WAP vilket leder till att det blir oberoende av den bärande tekniken och inte begränsat till endast GSM- eller WCDMA-nät [13,14].

Vid en närmare jämförelse mellan SMS och MMS finns både likheter och skillnader. Något de har gemensamt är ``store and forward" -principen. Följaktligen skickas leveransrapporter på meddelanden också i MMS. En av de största skillnaderna är att MMS-meddelanden skickas på datakanalen istället för signaleringskanalen. Fördelen med att använda signaleringskanalen är som tidigare nämnts att överföringen kan ske oberoende övrig data. Att MMS använder datakanalen ses dock inte som något problem eftersom ett genombrott väntas först efter lanseringen av tredje generationen mobiltelefoni (3G) där överföringshastigheten inte är en begränsande faktor. Vidare är SMS en löst specificerad standard som har gett upphov till en fragmenterad marknad av leverantörer där varje SMSC-tillverkare gjort sitt eget API. Med MMS används istället för en SMSC en handfull oberoende plattformar som använder etablerade protokoll vilket leder till ökad konkurrens och större flexibilitet [13,14].

5.7 Gränssnitt (API) för SMS tjänster

En mängd protokoll för SMS-hantering finns i dagsläget tillgängliga. Antingen kan man använda ett gränssnitt (API) direkt mot en operatör eller använda sig av ett gränssnitt skapat av en tredje part. De olika gränssnitten kan fungera på helt olika sätt och erbjuda mer eller mindre funktionalitet. Protokollet definierar enbart vad kommunikationen ska bestå av och hur kommunikationsflödet ska hanteras vid SMS-hantering. Protokollet förutser att bärare av datorkommunikationen mellan den externa applikationen och SMSC finns tillgänglig. Vilken bärare som används vid kommunikation med ett gränssnitt varierar, men vanligtvis används fasta förbindelser över Internet.

Vid SMS-hantering direkt mot en operatör tar applikationen kontakt med en meddelandecentral (SMSC), som befinner sig på en specifik Internet-adress, och berättar sedan vad den vill åstadkomma. Ingen övrig access med GSM-nätet behövs. Exempel på funktionalitet som kan erbjudas av SMSC och det API man använder är att utskick av SMS kan göras från applikationen, att SMS-meddelanden kan mottas från GSM-telefoner eller andra externa applikationer, via SMSC (sk. mobil-originerade SMS), samt att status för utskick av SMS från applikationen kan erhållas från SMSC.

Vid användning av ett API mot en tredje part tar applikationen först kontakt med tredje partens applikation via Internet. Tredje partens applikation sköter sedan kommunikationen mot operatörens meddelandecentral. Exempel på befintliga protokoll som finns tillgängliga är: External Machine Interface (EMI) [15], Universal Computer Protocol (UCP) [16], Computer Interface to Message Distribution (CIMD, CIMD2) [17] och Short Message Peer to Peer (SMPP) [18]. Nedan presenteras det gränssnitt, CIMD2, som användes vid utveckling av den SMS-funktionalitet som skapades för det utvecklade meddelandesystemet.

5.8 CIMD2 protokollet

CIMD2-protokollet är ett protokoll för överföring av SMS textmeddelanden till mobiltelefoner. Protokollet är gjort för att kommunicera med Nokias SMSC. Europolitan är en operatör som använder sig av Nokias SMSC. För att kunna skicka SMS via Europolitan så måste man ansöka om att få ett konto i deras SMSC. Uppkoppling till Europolitans SMSC sker via TCP/IP sockets. Inloggnings-förfarandet påbörjas med att en applikation, Short Message Entity (SME), öppnar upp en socket mot Europolitans SMSC. För varje konto har en applikation endast möjligheten att ha en samtidig socket kopplad mot Europolitans SMSC.

Med Nokias SMSC och CIMD2 protokollet är det möjligt att från en applikation både skicka och ta emot SMS meddelanden. SMS meddelandena skickas till en brevlåda som varje konto får hos Europolitan. Varje applikation får även ett unikt nummer i Europolitans SMSC, vilket innebär att applikationen kan kolla om den har något SMS meddelande att hämta. Nedan beskrivs några av CIMD2 protokollets viktigaste funktioner. För ytterligare information om funktioner och operationer i CIMD2 protokollet se bilaga A [17].

5.8.1 Operationer från SME till SMSC

Login (01)	Innan något annat kan göras måste en inloggning ske.
Logout (02)	Logoutoperationen indikerar slutet på en session med en SMSC men avslutar ej kopplingen till den. Efter logoutoperationen kan kopplingen avslutas.
Submit message (03)	Denna operation används för att skicka själva SMS meddelandet till mobiltelefoner eller andra applikationer. I submitfunktionen kan en förfrågan om statusrapporter för det skickade meddelandet begäras.
Enquire message status (04)	Den här funktionen används av en SME för att göra en förfrågan om en statusrapport om ett tidigare skickat SMS meddelande.
Delivery request (05)	Den här funktionen används av en SME för att hämta ett SMS meddelande.
Cancel message (06)	Den här funktionen används av en SME för att avbryta ett SMS meddelande den har sänt.
Set (08)	Den här funktionen används av en SME för att ändra vissa parametrar i gränssnittet.
Get (09)	Den här funktionen används av en SME för att ta reda på vilka värden som parametrar i gränssnittet har.

5.8.2 Operationer från SMSC till SME

Deliver message (20)	Den här funktionen används av en SMSC för att automatiskt skicka ett SMS till en SME.
Deliver status report (23)	Den här funktionen används av en SMSC för att sända en statusrapport som beskriver nuvarande status av ett SMS meddelande skickat av en SME.

5.8.3 Vanliga operationer

Alive (40)	Den här funktionen kan användas av båda parter för att testa om kopplingen är uppe.

5.8.4 Felmeddelanden

General error response (98)	Den här funktionen användas av en SMSC för att svara på ogiltiga operationer som sänts av en SME.
Nack (99)	Ett NACK meddelande används för att avvisa en operation eller meddelande som har felaktig kontrollsumma eller sekvensnummer. Sekvensnumret för ett NACK meddelande är alltid det förväntade sekvensnumret.

5.8.5 Meddelandets format

För varje operation består meddelandet av en header, ett datafält och en trailer. All data som skickas mellan en SMSC och en SME skickas som byteströngar. Varje operation har ett nummer eller kod som berättar vilken typ av meddelande det är, t.ex. har login meddelanden som sänds från en SME operationskoden 01.

5.8.6 Header

Headern ser ut som följande

<stx>ZZ:NNN<tab>

Där <stx> är ett starttecken som består av en enda byte med decimalvärdet 2. Fältet ZZ definierar operationskoden som meddelandet har och består av två bytes innehållande ASCII värdena för siffrorna 0-9 dvs 48-57. Operationskoden säger vilken typ av meddelande det är. Fältet NNN representerar meddelandets paketsekvensnummer. Fältet består av tre byte innnehållande ASCII värdena för siffrorna 0-9 dvs 48-57.

Paketsekvensnumren för meddelanden från en SME till en SMSC är ojämna nummer från 1 till 255. Numren ökar alltså med 2 per gång och när de når 255 börjar det om på 1 igen. Alla svar från en SMSC skickas tillbaka med samma sekvensnummer som original-meddelandet hade. Meddelanden från en SMSC till en SME:n använder jämna paket-sekvensnummer som börjar på 0. Även dessa nummer ökar alltså med 2 per gång tills dess att 254 nås, då börjar det om från 0 igen. Fälten ZZ och NNN åtskiljs av en byte som är ASCII värdet för ett kolon, som är 58. Headern avslutas av en byte med ASCII värdet för <tab>, som är 9. Exempel på en header:

<stx>01:001<tab>

blir i byte

2 48 49 58 48 48 49 9

5.8.7 Datafältet

Datafältet består av en lista av parametrar som avslutas med <tab> tecknet. Varje parameter har följande utseende:

PPP:operationens meddelande<tab>

PPP indikerar vilken parameterkod det är, och består av 3 bytes innehållande ASCII värdena för siffrorna 0-9. Efter skiljetecknet, kolon, är värdet av parametern skrivet i varierande längd av bytes, meddelandet avslutas med <tab>(9).

De olika operationerna som t.ex. login har olika parametrar som kan användas i meddelandet. Generellt kan man säga att varje parametervärde består mestadels av ASCII värden för siffror och bokstäver, men andra tecken kan också förekomma. De reserverade värdena 0x00(NULL), 0x02(STX), 0x03(EXT) och 0x09(TAB) kan inte användas i något parametervärde. Vilka parametrar som är tillåtna bestäms av vilken sorts meddelande det är, dvs vilken operationskod meddelandet har. Login meddelandet kan ha användarnamn och lösenord men inte så många andra parametrar. I vilken ordning parametrarna anges är valfritt och det finns ingen gräns för hur många parametrar ett meddelande kan ha.

Exempel av lösenordsparametern

011:lösenord<tab>

48 49 49 58 108 10 115 101 110 111 114 100 9

Exempel av userdata parametern

033:Hej !<tab>

48 51 51 58 72 101 106 32 33 9

5.8.8 Trailer

Den avslutande trailern ser ut så här.

CC<ext>

Där CC är två bytes som består av kontrollsumman för meddelandet och <ext> är en byte som är ett avslutningstecken, ASCII värdet 3. Det är valfritt att använda kontrollsumman, och om valet är att inte använda den består trailern bara av avslutningstecknet. Kontrollsumman används för att kontrollera att meddelandet kommer fram oförvanskat och inte blivit korrupt under sändning. Om kontrollsumman är inkorrekt så kommer en SMSC att skicka tillbaka ett NACK med det inkorrekta meddelandets sekvensnummer.

Kontrollsumman räknas ut genom att ASCII värdena för alla tecken i meddelandet fram till och med sista <tab> tecknet adderas ihop. Sedan tar man den summan modulo 256. Talet som då erhålls är ett tal mellan 0-255. Detta tal utgör då kontrollsumman när det skrivs hexadecimalt som två ASCII bytes. För hexkod är basen i talsystemet inte 10 utan 16. Talen som då används är 0 till och med F. Talet 255 blir FF hexadecimalt.

5.8.9 Exempel: Skicka ett SMS

Det första som händer när en applikation (SME) kopplar upp sig mot en SMSC, är att applikationen mottar en informationstext som kan se ut så här.

”CIMD2-B ConnectionInfo: SessionId = 16965 PortId = 4 Time = 000718113629 AccessType = TCPIP_SOCKET PIN = 685720<lf>”

Informationstexten börjar med information om aktuellt protokoll, i detta fall CIMD2-B. Resten är information om den just skapade kopplingen. Sist kommer ett radbrytning-stecken, ASCII tecknet 10. När en applikation fått informationssträngen är det dags att logga in. För att logga in så skickas strängen för ett login meddelande som kan se ut så här:

<stx>01:001<tab>010:Loginnamn<tab>011:Lösenord <tab><CS><ext>

Login meddelandet inleds med <stx> som är startkoden. Efter startkoden följer operationskoden 01, ett kolon samt sekvensnumret 001. Tecknet <tab> skiljer trailern från datafältet som börjar med parameterkoden 010 vilket är koden för loginnamn. Efter kolonet kommer själva loginnamnet. Ett <tab> tecken skiljer de olika parametrarna åt. Nästa parameterkod är 011 som är parameterkoden för lösenordet. En avslutande <tab> skiljer sista datafältet från trailern. Trailern består av checksumman som skrivs i hexkod med stora bokstäver och avslutas med avslutningstecknet. Positivt login svar från en SMSC kan se ut så här:

<stx>51:001<tab>3C<ext>

Meddelandet börjar som vanligt med en startkod följt av en operationskod för loginsvar som är 51. Efter kolonet kommer vilket sekvensnummer svaret gäller, i detta fall 001. Sist efter <tab> tecknet kommer trailern. När kopplingen är uppe och inloggningen är klar kan andra meddelanden börja skickas mellan en applikation och en SMSC. För att skicka ett sms meddelande kan en applikation exempelvis skicka detta meddelande till en SMSC.

<stx>03:003<tab>021:+46702334554<tab>033:Hej<tab><CS><ext>

Operationskoden för sända SMS är 03, sedan följer sekvensnumret. Numret 021 är parameterkoden för destinationsadressen och efter kolonet följer mottagarens mobilnummer. Sedan kommer parameterkoden för användardata, 033. Med användardata menas själva innehållet i det textmeddelande som ska skickas. Positivt svar från en SMSC på föregående operation kan se ut så här:

<stx>53:003<tab>021:+46702334554<tab>060:9640907476 53<tab><CS><ext>

53 är operationskoden för svar på ett skickat SMS meddelande. 021 är som tidigare destinationsadressen. 060 är koden för tidsstämpeln och efter kolonet kommer tiden i millisekunder när meddelandet mottogs av aktuell SMSC. Ett logout meddelande kan skickas innan uppkopplingen tas ner. Exempel på logoutmeddelande:

<stx>02:005<tab>3C<ext>

02 är operationskoden för ett logoutmeddelande, 005 är sekvensnumret. Logout-meddelanden har inte några parameterar utan avslutas med en trailer. Positivt logoutsvar från en SMSC kan se ut så här:

<stx>52:005<tab>41<ext>

Logoutsvaret har inte heller några parametrar utan bara operationskoden 52 följt av sekvensnumret på mottaget logoutmeddelande. Sist kommer trailern.

6. Genomförandebeskrivning

Här beskrivs genomförandet av de olika stegen i detta examensarbete. I november år 2000 påbörjades arbetet vid IBFS lokaler i Uminovahuset på Tvistevägen 46 i Umeå. Under några möten diskuterades behovet av ett meddelandesystem för företaget och ett projekt initierades för att ta tag i uppgiften. Tre projektmedlemmar utsågs, jag, Alexander Sehlin och Peter Sönnergren. Denna uppgift godkändes som den praktiska delen i mitt examensarbete när specifikationen för examensarbetet lämnades in till institutionen för datavetenskap vid Umeå universitet.

Vi valde att utveckla detta system med hjälp utav programmeringsspråket Java. Detta val gjorde vi bland annat för att det var det enda språk som vi alla tre hade erfarenhet utav. Ett annat alternativ som vi övervägde var C++ men eftersom vi ville att systemet skulle vara plattformsoberoende var Java ett bättre val.

Vi utvecklade alla på datorer med Windows NT som operativsystem. Vår utvecklingsmiljö bestod i övrigt utav bland annat en enkel editor, Ultra Edit, databaserna Microsoft Access och Oracle, samt versionshanteringssystemet CVS. Den produktionsmiljö där vi skulle köra det färdiga systemet på var i första hand Unix datorer med Solaris som operativsystem. Detta var den produktionsmiljö som användes på företaget samt den miljö som de flesta tilltänkta företagskunder till detta system använde sig utav.

6.1 Kravspecifikation

Första tiden ägnades åt vilka krav vi skulle ha på systemet. Det som kändes lite konstigt med detta projekt var att uppdragsgivarna, företagsledningen i detta fall, egentligen inte visste så mycket om det system de ville ha. Detta innebar att vi som ingick i projektet till stor del fick bestämma vilka krav som skulle ställas på systemet och utforma kravspecifikationen nästan helt på egen hand.

För att användare skulle kunna kommunicera med systemet och exempelvis specificera vilken information de är intresserade av, bestämdes att vi skulle göra åtminstone en JSP/HTML klient. Denna klient skulle vara en del i en större JSP/HTML klient till företagets aktiehandelssystem ROX (Realtime Order eXecution). Denna större klient höll redan på att utvecklas i ett projekt som pågick parallellt med detta projekt. Om vi hann skulle vi också försöka göra en JSP/WML klient till mobiltelefonen Ericsson R380. På samma sätt skulle denna klient också vara en del i en större JSP/WML klient till ROX som även denna höll på att utvecklas parallellt med detta projekt.

Vi graderade kraven som ställdes på systemet i tre nivåer beroende på hur viktigt det var att kravet blev uppfyllt i den första versionen utav systemet. Vi specificerade krav som rörde exempelvis systemets businesslogik, informationskällor, kommunikationskanaler, modularitet, beroenden, interna och externa protokoll samt loggning och användarhantering. Ett krav av högsta prioritet var att systemet skulle kunna ta emot marknadsdata från företagets aktiehandelssystem ROX. Ett annat krav som också fick högsta prioritet var att systemet skulle kunna förmedla information till mottagare genom att skicka ut SMS meddelanden. Tillslut godkändes kravspecifikationen av alla parter. En fördel med att skriva sin egen kravspecifikaitonen är att man kan göra preliminära tester och fundera ut vad som är möjligt att göra och på så sätt inte sätta orealistiska krav. En nackdel är att det tar lite tid att skriva den.

6.2 Förstudier och fördjupning

Parallellt med kravspecifikationen började vi ägna oss åt förstudier av olika tekniker och byggstenar som vi tänkt använda oss utav. Denna fördjupning forsatte sedan långt in i projektet. Saker som vi tittade närmare på och utredde var exempelvis JDBC, SQL, Oracle, XML, JSP, WML och olika SMS protokoll. Den information vi behövde inhämtades från Internet samt olika böcker.

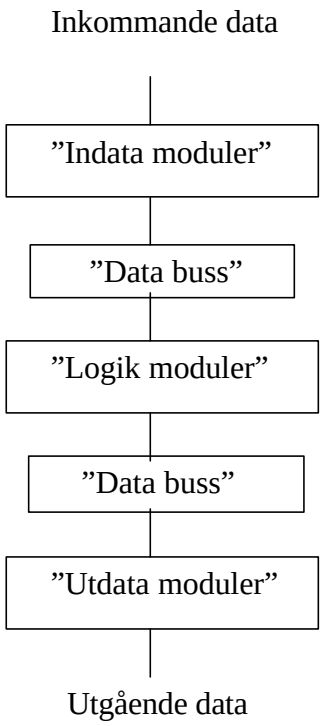
Vi bestämde oss bland annat för att basera systemets interna protokoll på XML. XML valdes fört att det är lättläst och kraftfullt. Valet av SMS protokoll föll tillslut på Nokias CIMD2 protokoll som verkade vara enkelt och effektivt. Europolitan var den operatör i Sverige som använde sig utav detta protokoll för hantering av SMS. Vi kontaktade dem och fick ett testkonto samt inloggningsuppgifter mot deras SMS central. Till en början strulade det en del, men med viss support från Europolitan kom vi igång att jobba mot deras system. Testkontot kostade 2500 kr i fast kostnad plus 500 kr i månaden och 0.95 kr per skickat SMS meddelande. För att kunna bygga en klient till mobiltelefonen Ericsson R380 så skaffade vi oss telefoner samt ett avtal med Telia för WAP tjänster via GSM nätet. Vid denna tidpunkt fanns inte GPRS tjänster tillgängliga.

6.3 Design

Innan vi satte igång med implementationen av systemet hade vi flera möten och diskussioner angående design utav systemets ingående delar. På detta företag användes vid denna tidpunkt inga speciella verktyg vid systemdesign. Detta innebar

att vi med enkla medel gjorde minimala men effektiva designskissar över hur vi ville att systemet skulle se ut.

Ett krav vi hade på systemet var att det skulle vara mycket flexibelt på så sätt att det skulle vara lätt att byta ut, lägga till samt ta bort komponenter i systemet. För att uppfylla kravet på flexibilitet införde vi i designen byggstenar som vi kallade moduler och bussar. Vi kom fram till en design där hela systemet skulle byggas upp dynamisk vid varje uppstart. Så lite som möjligt skulle vara förutbestämt och det skulle vara enkelt att byta ut komponenter mellan olika körningar. Vår lösning för att uppfylla denna design blev att alla komponenter i systemet specificerades i en XML fil. Vid uppstart av systemet kunde då de komponenter som specificerats i XML filen skapas dynamiskt och tillsammans bygga upp system med olika utseende vid varje uppstart. Något förenklat kan sägas att systemet byggdes upp utav tre lager av moduler med en buss mellan varje lager. Varje lager av moduler kunde innehålla ett godtyckligt antal moduler. Se figur 1 nedan.



Figur 1, Moduler och bussar

Det översta lagret av moduler tänkte vi skulle sköta extern kommunikation och hantera inkommande data som skulle omvandlas till meddelanden i systemets interna xmlformat innan de skickas till den första bussen. All businesslogik som systemet skulle ha tänkte vi att modulerna i det mittersta lagret skulle innehålla. Mittenmodulerna skickar sedan ned meddelandena på den nedre bussen när de är klara med sina uppgifter. Alla hantering för att skicka ut meddelanden från systemet på olika sätt via olika kanaler planerade vi att det understa lagret av moduler skulle ta hand om.

Bussarna designades för att vi behövde något som på ett dynamiskt sätt snabbt kunde fördela inkommande meddelanden till de rätta komponenterna. Den övre bussen skulle då fördela meddelanden till de moduler som hanterar businesslogik. Den undre bussen skulle fördela meddelanden till de olika modulerna som hanterar extern kommunikation för meddelanden som skickas ut från systemet.

För att systemets hantering av databaser skulle vara flexibel och utbytbar designades en separat komponent för detta ändamål. Denna komponent skulle exponera ett gränssnitt mot den eller de moduler i det mittersta lagret som behövde kommunicera med en databas. Denna lösning gör att ett byte av databas bara påverkar databas-komponenten och inte de övriga modulerna i systemet. I ett första skede använde vi en Microsoft Access databas för att testa den databasdesign som tagits fram. Ganska snabbt byttes accessdatabasen ut mot en Oracle databas.

Parallellt med designen av meddelandehanteringssystemet jobbade vi även med design av vissa delar i de två klientprojekt som också pågick på företaget. Dessa projekt som handlade om att ta fram två nya klienter till företagets aktiehandels system ROX. Den ena klienten var en JSP/HTML klient och den andra klienten var en JSP/WML/WAP klient för mobiltelefonen Ericsson R380.

När det gäller designen JSP/HTML klienten så fick vi mycket gratis av det som redan fanns på plats i detta projekt. JSP/WML klienten var vi däremot tvugna att designa själva från början till slut. Denna design handlade mest om hur vi skulle kunna visa mycket information på telefonens relativt lilla display samtidigt som klienten skulle vara användarvänlig. Slutligen enades vi om ett designförslag där alla menyalternativ låg som flikar i två lager i den övre delen av displayen. Det blev snyggt och visade sig också fungera bra.

6.4 Implementering

Eftersom vi alla hade jobbat med Java förut så var inte programmeringspråket något problem. Vissa delar av Java var vi dock något obekanta med och behövde där förbättra våra kunskaper. Ett exempel var Javas API för att kommunicera med databaser, JDBC, som vi fick läsa in oss på för att komma igång på detta område. Ett annat exempel var Javas API för att hantera och använda XML.

Förutom kopplingen till företagets aktiehandelssystem ROX så fanns ingen kod som vi kunde återanvända för att implementera meddelandesystemet. Vi delade upp implementeringen av systemets ingående komponenter mellan oss. Bra kommunikation gjorde att alla ändå hade bra kontroll på vad som hände på olika delar av projektet. Detta var nödvändigt för att få alla delar att passa ihop på det flexibla sätt vi hade som krav på systemet. Vi använde företagets kodmallar för projektet så att allas kod skulle se enhetlig ut för att det skulle vara lättare att förstå och sätta sig in i kod som man själv inte skrivit.

Implementeringen av själva meddelandesystemet tog längre tid än beräknat på grund av att vi inblandade också jobbade deltid på de två klientprojekten som pågick på företaget. Vid implementeringen av klienterna för meddelandesystemet kunde vi som tur var återanvända mycket av det som redan implementerats i de andra klient-projekten.

JSP/WML klienten var något svårare att implementera än JSP/HTML klienten. De båda klienterna hade dock väldigt mycket gemensamt. Något förenklat så behövde vi nästan bara ändra så att JSP sidorna genererade och skickade ut WML i stället för HTML. Naturligtvis stötte vi på olika typer av problem under vägen. Problem som rörde WML klienten var ofta lite svårare att komma till rätta med eftersom ingen hade någon som helst erfarenhet av att arbeta med klienter för mobiltelefoner.

Implementeringen utav SMS modulen i systemet var en stor och tidskrävande del utav projektet. Efter mycket studerande av protokollet CIMD2 satte vi igång med kodandet. Det tog flera veckor innan vi fick tillgång till vårt testkonto hos Europolitan. Detta gjorde att vi skrev en minimalistisk SMS central parallellt med vår SMS modul för att kunna testa det vi gjorde steg för steg. Naturligtvis fungerade allt inte direkt när vi för första gången kopplade upp oss mot den riktiga SMS centralen. Efter någon mindre justering lyckades vi dock ansluta till vårt konto. Sedan kunde vi i allt snabbare takt implementera de funktioner vi ville att vår SMS modul skulle ha. Vi implementerade inte hela protokollet fullt ut. Lyckan var stor när vi hörde de första signalerna från våra mobiltelefoner som indikerade att meddelandesystemets SMS meddelanden nådde sina mottagare.

6.5 Testning

Testning pågick naturligtvis mer eller mindre under alla faser utav projektets gång. Vi upptäckte bland annat stora skillnader i trådhantering på olika operativsystem. Vi löste dessa problem genom att göra det enkelt att justera vissa inställningar för systemet beroende på vilken plattform och operativsystem det skulle köras på. Detta var dock inget allvarligt problem eftersom väldigt hög prestanda inte var ett krav av högsta prioritet i den första versionen utav systemet.

Ett annat prestandarelaterat problem som till en början var allvarligt var det att vår kommunikation med databasen Oracle gick mycket långsamt. Detta blev bättre och nådde en tillfredställande nivå efter ett antal optimeringar av kommunikationen med databasen.

SMS modulen och de båda klienterna, framförallt WML klienten, krävde en hel del testning. Den mesta testningen av dessa komponenter utfördes dock under utvecklingsfasen eftersom det var nödvändigt för att komma vidare med implementeringen i de flesta fall. När testfasen tillslut var avklarad fungerade systemets alla komponenter tillfredställande och det kändes mycket bra för oss som jobbat med projektet.

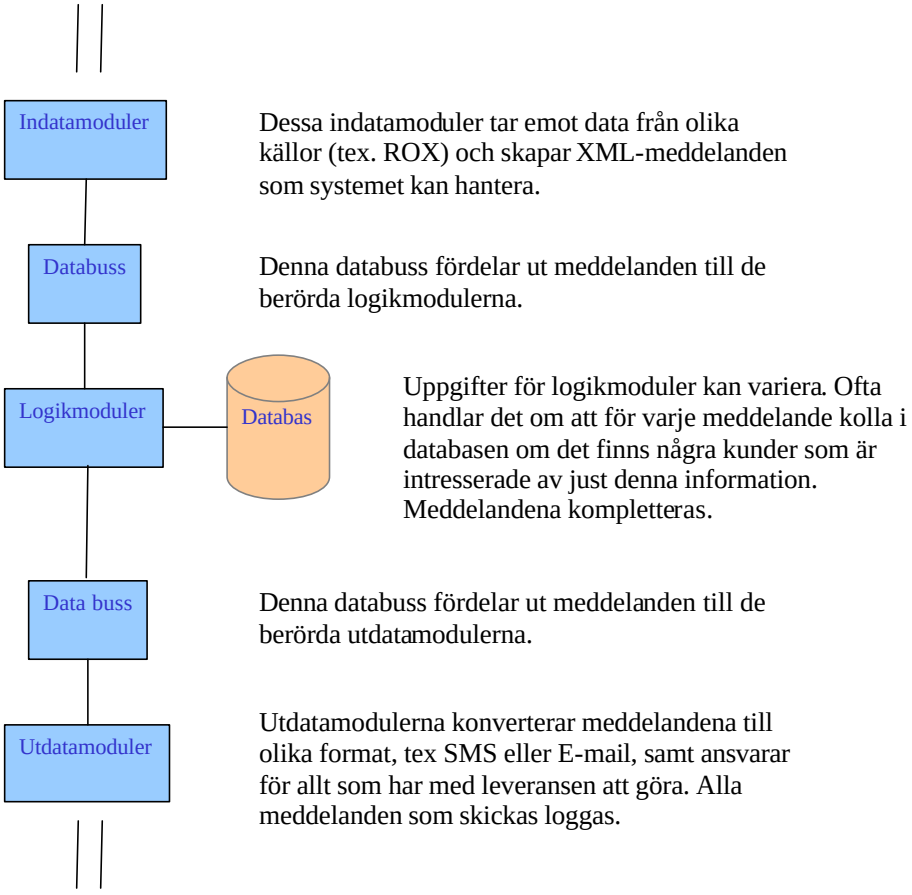
7 Resultat

För att utveckla detta meddelandesystem har det krävts kunskap om många olika teknologier och områden inom programvarutveckling. Ett av målen har hela tiden varit att systemet skulle vara så flexibelt och modulärt som möjligt. Det tycker vi att vi lyckats med. Fördjupningsstudier, analys och design samt dokumentation var tidskrävande, men implementeringfasen tog nog mest tid.

Eftersom detta examensarbete utfördes vid ett privat företag där all information är konfidentiell kan tyvärr inte dokument eller källkod som producerats under projektets gång redovisas i denna rapport. Presentationer av olika delar av systemets arkitektur, meddelandeflöde, användning och administration följer nedan.

7.1 Arkitektur

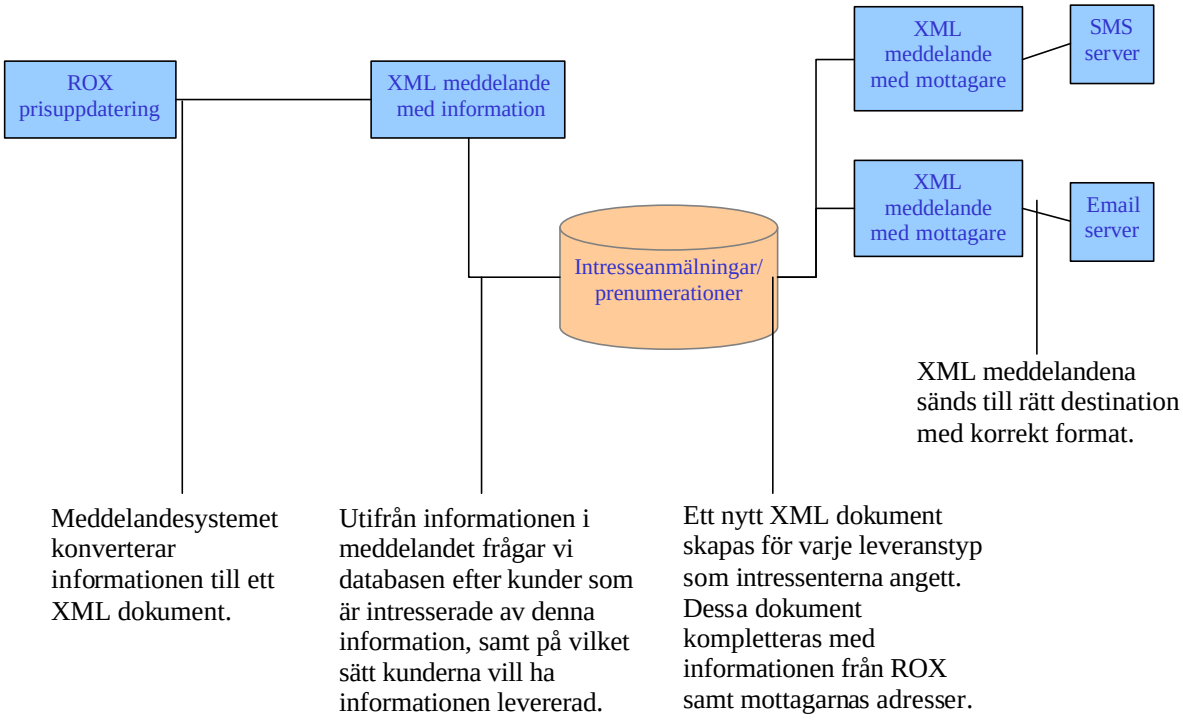
I figur 2 nedan visas en något förenklad bild utav meddelandesystemets arkitektur, samt förklarande text till de olika delarna. Något förenklat kan sägas att systemet består av tre lager av moduler, samt en databuss mellan varje lager. All business-logik sköts av modulerna i det mittersta lagret som också använder en databas. Systemet kan startas med ett valfritt antal moduler i varje lager.



Figur 2, Arkitektur

7.2 Meddelandeflöde

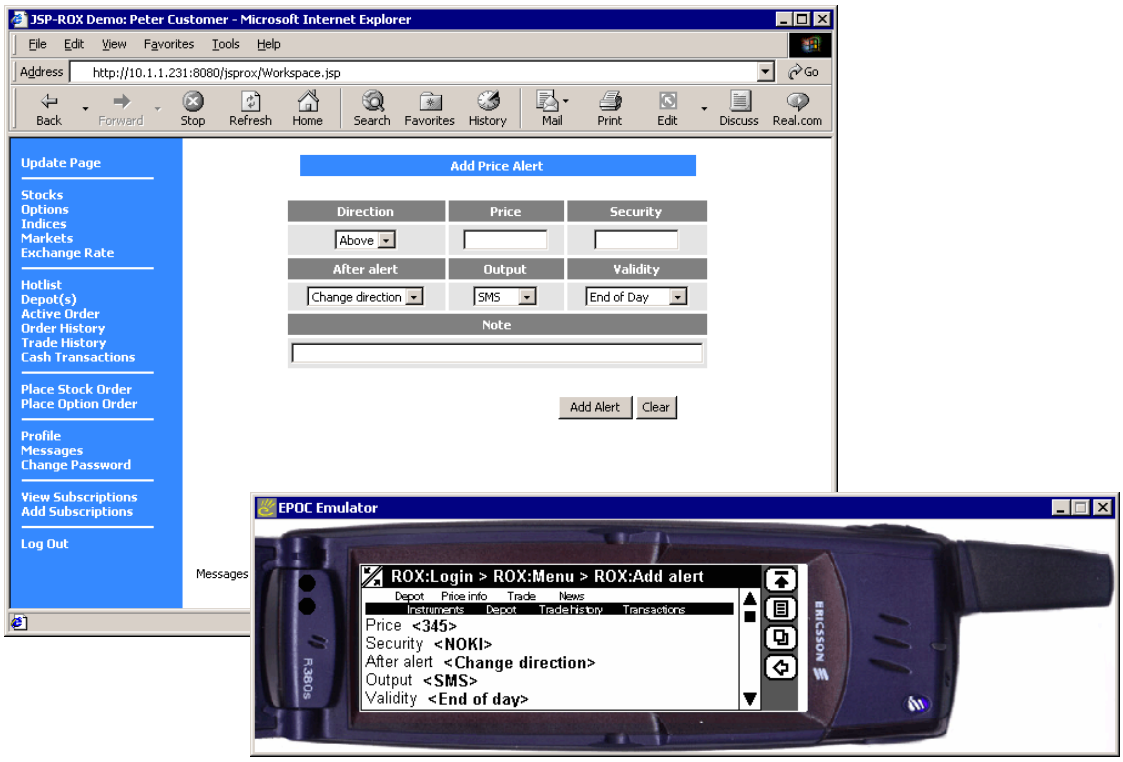
I figur 3 nedan visas en något förenklad bild utav meddelandeflödet i systemet, samt förklarande texter. Detta exempel visar hur meddelandeflödet ser ut när aktiehandelssystemet ROX skickar en prisuppdatering på ett instrument till meddelandesystemet. Meddelandet kompletteras sedan med information från databasen och skickas slutligen till användarna via olika externa system.



Figur 3, Meddelandeflöde

7.3 Användargränssnitt och administration

För att användare på ett enkelt sätt skulle kunna använda systemet gjordes två olika klienter, en JSP/HTML klient och en WAP/WML klient för mobiltelefonen Ericsson R380. Via dessa klienter kunde användarna bland annat själva specificera och administrera vilken information de var intresserade av och vid vilka tillfällen de ville få information från meddelandesystemet. Klienterna till meddelandesystemet byggdes ihop med företagets klienter till aktiehandelssystemet ROX. I figur 4 nedan visas hur det kan se ut när en användare håller på att lägga in en intresseanmälan, prenumeration via de två klienterna. I denna figur kan vi se att användaren vill bli informerad när en utvald aktie når en specifik börskurs.

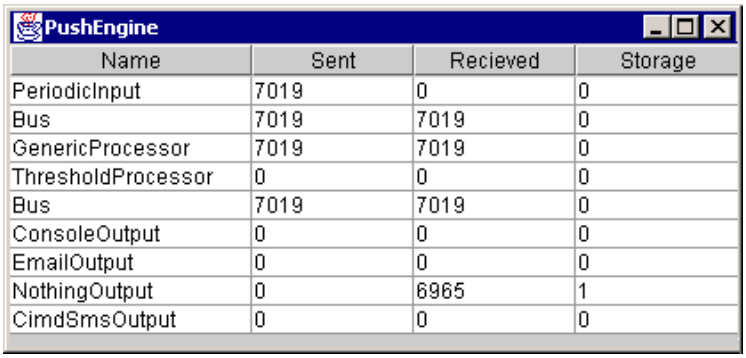


Figur 4, Klienter

7.4 Statuskontroll

För att kunna kontrollera och övervaka meddelandesystemet byggdes en enkel grafisk applikation, se figur 5. Med denna applikation kan en administratör från vilken dator som helst enkelt ansluta till ett meddelandesystem som körs på en godtycklig dator, förutsatt att båda datorerna är anslutna till internet.

Applikationen visar status för alla moduler och bussar som finns i aktuellt meddelandesystem. Uppdateringen av data till applikationen sker två gånger per sekund. Förutom att kontrollera systemets prestanda och status kan denna applikation även användas som ett verktyg för att hitta eventuella flaskhalsar i systemet. Antalet skickade SMS och emailmeddelanden kan också enkelt kontrolleras. Se exemplet i figur 5 nedan.



Name	Sent	Recieved	Storage
PeriodicInput	7019	0	0
Bus	7019	7019	0
GenericProcessor	7019	7019	0
ThresholdProcessor	0	0	0
Bus	7019	7019	0
ConsoleOutput	0	0	0
EmailOutput	0	0	0
NothingOutput	0	6965	1
CimdSmsOutput	0	0	0

Figur 5, Monitor GUI

7.5 Demonstration av systemet

Resultatet av vårt arbete tillsammans med företagets övriga produkter visades på SIA (Securities Industry Association) mässan i New York, juni 2001. Meddelandesystemet tillsammans med företagets aktiehandelssystem ROX och WAP/WML klienten för mobiltelefonen Ericsson R380 visade sig bli en kraftfull kombination.

Under demonstrationen visades ett scenario där en potentiell aktiehandlare via den mobila klienten först specificerade sitt intresse för när en specifik aktie nådde en förutbestämd prisnivå. En simulator till ROX gjorde så att denna nivå nåddes inom några minuter. Meddelandesystemet genererade då direkt ett SMS meddelande till samma mobiltelefon. Meddelandet innehöll förutom relevant information om aktuell händelse en direktlänk till den funktionalitet i klienten som används för att handla med aktuellt instrument. Alla värden i klienten var då förifyllda med uppdaterad börsinformation och användarens inställningar. Resultatet blev att användaren endast behövde klicka på länken, ange sitt lösenord och svara på en enda bekräftelsedialog för att direkt kunna agera genom att köpa eller sälja aktuellt instrument. All kommunikation gick mellan mobiltelefonen i USA, via GSM-nätet och internet, och våra datorer i Umeå där meddelandesystemet, webservern och ROX kördes.

8. Slutsatser

Detta examensarbete var rätt omfattande och tog därför längre tid än beräknat. Tack vare att vissa delar i det slutgiltiga systemet, exempelvis klienterna, inte behövde utvecklas från grunden blev situationen hållbar. Uppgiften i detta examensarbete och projekt var att skapa ett modulärt meddelandesystem. Det system som skapades fungerade enligt uppsatta mål, blev flexibelt och hade godtagbar prestanda. Vi hann även med att visa upp systemet på en stor mässa i USA med bra resultat. Allt detta sammantaget visar på att vi lyckats slutföra uppgiften med tillfredställande resultat.

Naturligtvis finns det många saker som kan förbättras med systemet, och funktionalitet som skulle kunna läggas till. Några av dessa berör SMS funktionaliteten. Exempelvis vore det önskvärt om meddelandesystemet kunde stödja flera olika SMS protokoll och operatörer. Mer avancerad kontroll över olika parametrar vid sändning av SMS skulle också vara användbart. Då skulle systemet, beroende på det aktuella meddelandets informationstyp, till exempel kunna styra hur länge en SMSC ska försöka skicka ut meddelandet till icke tillgänglig kund.

Andra saker som vore önskvärt är exempevis stöd för ytteligare in- och ut-kanaler. Systemet skulle kunna ta emot exempelvis nyheter från en Reuters feed och skicka ut specifika nyheter till kunder som anmält intresse. En tänkbar ny utkanal skulle kunna vara att leverera meddelanden till nya generationer av mobiltelefoner med så kallad WAP-Push teknik.

Säkerhet och prestanda är områden som vi inte hann titta på och arbeta med speciellt mycket. Här skulle vi behöva göra undersökningar och utredningar över vad som är

godtagbara nivåer. Därefter skulle vi kunna kontrollera hur vi ligger till med nuvarande lösning genom olika tester, och förbättra systemet där eventuella brister upptäcks. Systemet skulle även behöva funktionalitet för att stödja flera olika databaser. Till sist kan sägas att det naturligtvis även vore bra att ha klienter som fungerar med andra webbläsare och mobiltelefoner än de som stöds idag.

9. Tack

Utan hjälp och stöd från ett antal personer skulle nog inte detta arbete ha kunnat genomföras. Därför skulle jag här vilja tacka de personer som bidragit på olika sätt under genomförandet av detta examensarbete. Jag vill tacka min interna handledare Jerry Eriksson för råd och stöd under rapportskrivandet. Sedan vill jag naturligtvis tacka företaget där jag genomfört mitt examensarbete, Interbizz Financial Systems. Där vill jag först och främst tacka de två personer jag arbetat tillsammas med i detta projekt, Alexander Sehlin och Peter Sönnergren. Ett tack går även till dessa personer inom företaget : Anders Brändström, Niclas Holmgren, Stefan Håkans, Erik Kostamo, Jonas Kröger, Ingvar Lundqvist, Tommy Myllymäki, Tomas Toyrä och Mats Wikström. Tack!

10. Referenser.

Alla Internetreferenser besöktes senast i maj 2004.

[1] Schach S R. *Classical and object-oriented software engineering, third edition*. Irwin. 1996

[2] Elmasri R, Navathe S B. *Fundamentals of Database Systems, third edition*. Addison-Wesley. 2000.

[3] Sun Microsystems. *The Java Web Services Tutorial*. Sun Microsystems. 2003. (http://java.sun.com/webservices/docs/1.3/tutorial/doc/index.html)

[4] Seth White, Maydene Fisher, Graham Hamilton, Rick Catell, and Mark Hapner. *JDBC™ API Tutorial and Reference, Second Edition: Universal Data Access for the Java 2 Platform.* Addison Wesley Longman, Inc. June 1999.

[5] Oracle Corporation. *Oracle 8i Enterprise Edition, Technical Data Sheet*. Oracle Corporation. 1999. (http://otn.oracle.com/products/oracle8i/pdf/8i_ent_ds.pdf)

[6] Brian Randell and Aaron Skonnard. *A Guide to XML and Its Technologies, DevelopMentor*. 1999. (http://msdn.microsoft.com/archive/default.asp?url=/archive/en-us/dnarxml/html/xmlguide.asp)

[7] Buckingham S. (2000). *An Introduction to SMS*. Mobile Lifestreams Ltd. (http://www.mobilegprs.com/wp/wp2.htm)

[8] ETSI. *Technical realization of the Short Message Service (SMS) Point-to-Point (PP)*. ETSI. ETSI TC-SMG 03.40, v 5.3.0. 1996

[9] ETSI. *Technical realization of Short Message Service Cell Broadcast (SMSCB)*. ETSI. ETSI TC-SMG 03.41, v 5.3.0. 1996

[10] Sempere J G. *An overview of the GSM system.* Communications Division, Department of Electronic & Electrical Engineering, University of Strathclyde, Glasgow, Scotland. 2000. (http://www.comms.eee.strath.ac.uk/~gozalvez/gsm/gsm.html)

[11] Mobile Streams Ltd. *GPRS Introduction Zone*. Mobile Streams Ltd. (http://www.mobilegprs.com/gprs.asp?link=1)

[12] Simon Buckingham. *What is SMS?*. GSM World. 2000. http://www.gsmworld.com/technology/sms/intro.shtml

[13] Mobile Streams Ltd. *MMS Overview*. Mobile Streams Ltd. (http://www.mobilemms.com/mmsoverview.asp)

[14] Mobile Streams Ltd. *MMS FAQ*. Mobile Streams Ltd. (http://www.mobilemms.com/mmsfaq.asp)

[15] CMG Telecomunications & Utilities BV. *Short Message Service Centre, External Machine Interface, Specification*. CMG Telecomunications & Utilities BV. EMISPEC.312, v 3.1.2. 1999

[16] Telia. *Mobil Text företag, Handbok för applikationsutvecklare*. Telia, Stockholm. 2000

[17] Nokia Networks Oy. *Nokia Artus Messaging Centers, Computer Interface to Message Distribution, User's Guide*. Nokia Networks Oy. 2000.

[18] SMPP Developers Forum. *Short Message Peer to Peer Protocol Specification v3.4*. SMPP Developers Forum. 1999

[19] Svahn M. *Interface Description, HTTP/XML*. Wireless Maingate AB. MG000137 AU, A. 2000

[20] David Flanagan, (1997). *Java in a Nutshell, Second Edition*, ISBN 1-56592-262-X

[21] Mary Campione, Kathy Walrath, Alison Huml, and the Tutorial Team, *The Java Tutorial Continued: The Rest of the JDK*, ISBN 0-201-48558-3

11. Förkortningar

3G Tredje generationens mobila internet
3GPP 3rd Generation Partnership Project
API Application Programming Interface
ASCII, American National Standard Code for Information Interchange
CBS Cell Broadcast SMS
CIMD Computer Interface for Messaging Distribution
CEPT Conference of European Posts and Telecommunications
CVS Concurrent Versions System
EMI External Machine Interface
ETSI European Telecommunications Standards Institute
ER Entity Relationship
GPRS General Packet Radio Service
GSM Global System for Mobile Communication
GSM PLMN GSM Public Land Mobile Network
GUI Graphical User Interface
HTML HyperText Markup Language
IBFS Interbizz Financial Systems
IP Internet Protocol
ISDN Integrated Services Digital Network
ISO International Standards Organization
JDBC Java Database Connectivity
JSP Java Server Pages
MMS Multimedia Messaging Service
MO Mobile Originating
MS Mobile Station
MSC Mobile Service switching Centre
MT Mobile Terminating
PLMN Public Land Mobile Network
ROX Realtime Order eXecution (IBFS application)
SME Short Message Entity
SMG Special Mobile Group
SMPP Short Message Peer to Peer
SMS Short Message Service
SMSC Short Message Service Center
SMS-CB SMS Cell Broadcast
SMS-GMSC Short Message Service Gateway Mobile services Switching Centre
SMS-IWMSC Short Message Service Interworking MSC
SMS-PP SMS Point to Point
SQL Standard Query Language
TCP Transmission Control Protocol
UCP Universal Computer Protocol
UI User Interface
W3C World Wide Web Consortium
WAP Wireless Application Protocol
WCDMA Wideband Code Division Multiple Access
WML Wireless Markup Language
XML eXtended Markup Language

Bilaga A – CIMD2 User's Guide

Nokia Artus SMSC

Computer Interface to Message Distribution

User Guide

Contents

Introduction

This section introduces the Short Message Entity (SME) and the Short Message Service Centre (SMSC) architecturally, and explains how the interface between these two systems will be specified in the rest of this document.
The SME interface described in this document is called CIMD2 (Computer Interface to Message Distribution).

System overview

A Short Message Entity (SME) is interconnected through CIMD2 to the Short Message Service Centre (SMSC). The main purpose of this interconnection is to send short messages from the SMEs to the Mobile Stations (MS) and from the MSs to the SMEs. Other kinds of information can also be conveyed over the interconnection, e.g. status reports from the GSM network to the SMEs.
The system architecture discussed in this document thus consists of the SME and the SMSC, and the purpose of this document is to specify the interface between the two.
When a short message has been submitted to the SMSC using the CIMD2, the SMSC uses its own retry policy to attempt to deliver the message. If the delivery fails, the message is stored in the SMSC database until a delivery attempt succeeds. If a permanent error occurs, or the validity period of the message expires, the message is deleted.
The computer interface is suitable for client applications which send and retrieve short messages. The interface supports TCP/IP sockets, X.25 PAD and serial ports (modems).
The user always identifies himself in a "login". A login ID does not have to be unique, but can be a commonly known name used in public services (like when using premium charge modem lines).
The CIMD2 operations are specified in *sections 2* and *4* of this document. Each CIMD2 operation carries a number of parameters with it, i.e. data items specifying the subscriber, some facts about the operation itself, etc.
Section 3 introduces the question of *how* to communicate between the SME and the SMSC, i.e. the coding of information related to operations and parameters.
The parameters are specified in *section 5* of this document.

Note

In some cases the operator may want to restrict the use of some parameters. These are configured by the operator in the interface profiles.

In this document the operations and the parameters specified for each operation represent the maximum amount of information the SME or the SMSC may provide. It is very important to notice that in most cases it is not reasonable for the SME to send all the possible parameters.

CIMD operations

This section defines the operations between the SME and the SMSC. The parameters related to each operation are specified in *section 4*; the values of the parameters are specified in *section 5*.
The operations are divided into operations originated by the SME, operations originated by the SMSC, and operations that can be originated by both the SME and the SMSC.

SME types

When defining the operations the different SMEs might use, three basic types of SMEs were considered:

1. send-only SMEs
2. querying SMEs
3. receiving SMEs

A send-only SME can only submit short messages to the SMSC. The short messages are destined to MSs or other SMEs. Status reports of sent messages must be requested by the SME explicitly.
A querying SME does not receive anything from the SMSC automatically, but merely queries if there is something to be retrieved. The SME is typically connected to the SMSC every now and then to submit a message and may at the same time also check if there is something to be received. An example of this kind of SME is a PC-application with a modem connection to the SMSC. Status reports, if desired, must be requested by the SME explicitly.
A receiving SME is always ready to receive if the SMSC has something to send to it (i.e. short messages or status reports). A receiving SME can automatically receive messages stored in the SMSC, when logging in to the SMSC. An SME can also be set up in such a way that messages stored in the SMSC are not sent automatically when SME logs in. In such a case, the SME receives new incoming messages, but it has to query for the old messages stored in the SMSC.
The type of the SME must be specified before the SME may operate. The type, along with other information about the SME, is stored in the SMSC.

Operations from the SME to the SMSC

operation	definition
login	This operation is used by all SMEs before any other operations.
logout	This operation is used by all SMEs to indicate the end of the session. The logout operation does not release the connection, a disconnect must be explicitly performed after the logout.
submit	This operation is used by the SME to send short messages to MSs and/or other SMEs. In the submit operation, the SME can request transmission of status reports for the short message.

delivery request This operation is used by the SME to retrieve short messages.

cancel This operation is used by the SME to cancel short messages it has sent.

enquire message status

This operation is used by the SME to request a status report for a previously submitted short message.

set This operation is used by the SME to change the values of those interface parameters that it is allowed to change.

get This operation is used by the SME to inquire the values of the interface parameters.

Operations from the SMSC to the SME

deliver message This operation is used by the SMSC for automatically delivering a short message to the SME.

deliver status report This operation is used by the SMSC to send a status report describing the current status of a short message sent by the SME.

Common operations

alive This operation can be used by both entities to check whether the link is alive.

Operations supported by different types of SMEs

+ = supported
- = not supported

Operation \ SME type	Send-only	Querying	Receiving
Login	+	+	+
Logout	+	+	+
Submit	+	+	+
Enquire message status	+	+	+
Delivery request	-	+	+
Cancel	+	+	+
Deliver message	-	+	+
Deliver status report	-	-	+
Set parameters	+	+	+
Get parameters	+	+	+
Alive	+	+	+

CIMD protocol messages

Message format

Each message, operation or response consists of a header, data and trailer part of the message. The coding of these parts is explained below.

Note

Any data transmitted between packets can be ignored. This data can originate from modems, terminal drivers etc.

Header

The Header has the following format:

```
<stx>ZZ:NNN<tab>
```

Where <stx> is the start-of-text indicator which consists of a single byte containing the decimal value 2.
The field ZZ defines the operation code and consists of two bytes containing the ASCII characters of the digits 0 to 9 which range from 48 to 57.
The field NNN represents the message packet number which consists of 3 bytes containing the ASCII characters of the digits 0 to 9 which range from 48 to 57.
The ZZ and NNN fields are separated by one byte containing the ASCII code of a colon, which is 58. The header is terminated by one byte containing the ASCII code of <tab> which is 9
An example of the decimal values for each byte of a header is:

```
2 48 49 58 48 48 49 9
```

In the notation used in the rest of this document this becomes:

```
<stx>01:001<tab>
```

Data

The DATA fields consist of a list of parameters each terminated by the <tab> character. Each parameter field has the following format:

```
PPP:value of the parm<tab>
```

Where PPP indicates the parameter type and consists of 3 bytes containing the ASCII values of the digits 0 to 9 (values 48 to 57). After the single byte containing the ASCII value for the colon (58) the value of the parameter is coded with a variable number of bytes. The parameter is terminated by a single byte containing the ASCII value for tab (9).
The coding of the parameter value depends on the type of the parameter and is explained in *chapter 5*.
Generally speaking, all parameters consist of the ASCII equivalents of digits or the characters of the alphabet. The parameter for the user data (033) however can consist also of other characters which will enable the use of all characters of the default GSM character set. The reserved characters 0x00 (NUL) 0x02 (STX), 0x03 (ETX), 0x09 (TAB) are not allowed in any parameter.
The parameters allowed in a packet depend on the operation and sometimes the user profile. The order of parameters is free, and many parameters may be omitted.
An example of the decimal values of the bytes for a few parameters is listed below together with the notation used in this document.

```
Example of the password parameter (11):
48 49 49 58 83 101 51 114 83 116 9
011:SeCrEt<tab>

Example of the userdata parameter (33)
48 51 51 58 104 105 32 116 104 101 114 101 32 33 9
033:hi there !<tab>
```

Trailer

The format of the trailer of a packet is as follows:

```
CC<etx>
```

where CC consists of two bytes containing the checksum of the packet, and <etx> is a single byte containing the end-of-text character which has the value 3.
The usage of the CC field is optional, in which case the Trailer merely consists of the single <etx> byte.
An example of the decimal values of the bytes of a trailer is listed below, together with the notation used in this document.


```
51 49 3
3A<etx>
```

Examples

All parts combined gives the following typical message:

```
<stx>ZZ:NNN<tab>PPP:parameter1<tab>QQQ:parm2<tab><etx>
```

or when using real values for ZZ,NNN etc:

```
<stx>01:001<tab>010:wwwstat<tab>011:wwwstat<tab><etx>
```

or:

```
<stx>03:011<tab>021:123456789<tab>033:hi there<tab><etx>
```

Message packet numbering

All CIMD2 messages are assigned a packet number. This packet number is used for detecting duplicate packets or missing packets. The number is assign according to the following rules:

- Operations from the SME to the SMSC are assigned an odd packet number, starting from 1. Subsequent packet numbers are incremented by 2. After reaching 255 the number wraps back to 1 again. Using the coding as described before the 3 character NNN field this becomes for a list of SME originated packets:

  001, 003, 005, ... 253, 255, 001, 003, ...

- Operations from the SMSC to the SME are assigned an even packet number starting from 0. The following packet numbers are incremented by 2. After reaching 254 the number wraps back to 0 again. This gives us for SMSC originated packets:

  000, 002, 004, ...,252 , 254, 000, 002, 004, ...

- All response messages are assigned the same packet number as the request. Thus, the responses from the SME to the SMSC have even numbers and the responses from the SMSC to the SME have odd numbers.

Checksum

The checksum is an optional field, but when it is used it is calculated according to the following procedure:

1. At the beginning of the message, set the checksum to 0.

2. Retrieve the first byte of the message.
3. Add the value of the byte to the checksum.
4. Truncate the checksum so that it contains only the least significant byte.
5. If available, retrieve the next byte from the message and go to step 3. The process stops when 2 bytes further in the message the <etx> field is found.

The checksum calculation using the C language can be:

```
int GetChecksum( char *pstx, char *petx )
/************************************************/
/* pstx points to the <stx> in the message      */
/* petx points to the <etx> in the message      */
/************************************************/
{
     int checksum = 0;
     char *p = pstx;

     while (petx - p >= 2 ){
          checksum += *p;
          checksum &= 0xFF;
          p++;
     }
     return( checksum );
}
```

This means that all characters from the first character to the last character *before* the checksum characters are included in the sum. Thus, <stx> is the first character in the checksum calculation and the last <tab> before the checksum is the last character. The checksum characters and the <ETX> are *not* included in the calculation.
The underlined portion of the example message below indicates the characters included:

<stx>ZZ:NNN<TAB>PPP:value1<tab>QQQ:val2<tab>CC<etx>

The coding of the checksum value into the two bytes of field CC is done as follows. The most significant 4 bits of the checksum is coded in the first byte; the least significant 4 bits are coded into the second byte of the checksum field CC. The ASCII representation of the digits ‘0’ to ‘9’ and ‘A’ to ‘F’ are used for coding the hexadecimal value of the four bits into the message.
For example, if the checksum is 58 (decimal) which is 0x3A (hexa-decimal), most significant 4 bits give us the value 3, and the ASCII representation ‘3’ has the value 51 (decimal) or 0x33 (hex). The second value gets the value ‘A’ which is 65.
The use of the checksum is optional for the SME. The SMSC will always include a checksum to the packets it sends to the SME.

Response messages

The interaction between an SME and the SMSC involves the sending of request messages to which the other party responds with a response message. Depending on the situation the response message can be one of the following.

Positive response message

After processing the request message, the SME or SMSC sends a positive response message back. The operation code of the response packet is fixed to be 50 more than the operation code of the request packet. The packet number is the same as the request message.
A positive response message informs the initiator of the operation that the request message was received correctly and the operation was performed successfully.
In some exceptional cases (e.g. cancel message), the response message assures only that the request message was received correctly and the operation will be performed, but the result of the operation is not available.

Negative response message

If the request contains invalid parameters, or can not be performed, a negative response message is sent back to the initiator of the operation. Positive and negative responses have the same operation code and packet number, but the negative response message has an error code and optionally also an error text parameter. The usage of error texts is set in the interface profile.
A negative response message informs the initiator of the operation that the request message was received correctly, but the operation could not be performed successfully.

Nack message

A nack message is a special case which performs either of the following actions:

1. It can be used to trigger a re-transmission of the request message in case the checksum calculated by the receiver does not match the checksum found in the packet.
2. The second use of the nack-message is to indicate that the packet number used is not the expected number. The packet number of the nack-message always contains the expected packet number.

Note

The SMSC will never change the packet number because of the nack-message. The SMSC always assumes that the packet number confusion should be corrected by the SME

General error response message

This response message is used if an unknown operation has been received by the SMSC. It indicates the correct reception of a request with a correct checksum and packet number, but an illegal operation code (for example operation code 7).

Operation numbers

All available operations are listed below, showing both the request and the response operation code.

SME	SMSC
request	**response**
----->	<-----
login (01)	login response (51)
logout (02)	logout response (52)
submit message (03)	submit message response (53)
enquire message status (04)	enquire message status rsp (54)
delivery request (05)	delivery request response (55)
cancel message(06)	cancel message response (56)
set message (08)	set response (58)
get message (09)	get response (59)
alive (40)	alive response (90)
	general error response(98)
	nack (99)

SME	SMSC
response	**request**
----->	<-----
deliver message rsp (70)	deliver message (20)
deliver status report rsp (73)	deliver status report (23)
alive response (90)	alive (40)
nack (99)	

Operation details

This chapter defines the details of each operation listing the parameters and values possible. In the next chapter the formatting and value range for each parameters is presented.

Login (01)

A login operation must always be done before any other operations. The SMSC selects the interface profile based on the user identity given in the login operation.

Nbr:	Login parameters:	Presence
010	user identity	M

011	password	M

Login response parameters:
Positive response contains no parameters.
Negative response parameters:

900	error code
901	error text (optional)

A negative response may follow if the user is already logged in or the user identity or password is incorrect.

Logout (02)

The logout operation closes the session but not the connection. This allows e.g. SMEs connected via modems to login again without having to re-establish the modem connection to the SMSC. In most cases the SME can close the connection without performing an explicit log out.

Logout parameters:

Logout operation does not need any parameters.

Logout response parameters:

Positive response contains no parameters.

Negative response parameters:

900	error code
901	error text (optional)

Submit message (03)

Submit in its simplest mode just passes the short message text and destination address to the SMSC which takes care of delivery. There are, however, also some special features that may be requested with the submit operation, such as, first delivery time, message to many recipients, etc.
When the SME wants to submit a short message, it builds the message text and places it into the parameter user data in the submit operation. The text is sent with other necessary parameters to the SMSC. The SMSC then sends the message as such to the MS or other SME.
The submitted short message can be identified afterwards by using a time stamp generated by the SMSC (returned in a submit response) and the destination address.

Nbr	**Submit parameters:**	Presence
021	destination address (multi)	M
023	originating address	O
030	data coding scheme	O
032	user data header (bin)	O
033	user data *	O
034	user data binary *	O
050	validity period relative **	O
051	validity period absolute **	O
052	protocol identifier	O
053	first delivery time relative ***	O
054	first delivery time absolute ***	O
055	reply path	O
056	status report request	O
058	cancel enabled	O
064	tariff class	O
065	service description	O
067	priority	O

* either user data (033) or user data binary (034) field
** either relative or absolute validity period
*** either integer or absolute first delivery time
For using the user data parameters (032, 033, and 034), refer to the parameters description in *section 5*.
If an originating address is given in the submit message, it is regarded as a sub-address, i.e. it is added to the end of the address that is set in the interface profile.

Submit response parameters:

Positive response parameters:

021	destination address
060	service centre time stamp

Negative response parameters:

021	destination address *
900	error code
901	error text (optional)

* destination address included only if SMSC could interpret the address in the request correctly

Note

A submit to multiple destinations is responded to with a single submit response message. This submit response contains an element, formed of a destination address and a time stamp (positive response) or of a destination address and an error code (negative response), for each destination address in the submit message.

Example of a Submit operation to multiple destinations:

```
Submit Request --->
<STX>03:007<TAB>021:11111<TAB>021:22222<TAB>021:333<TAB
>033:UserData<TAB><ETX>

<--  Submit Response
<STX>53:007<TAB>021:11111<TAB>060:971107131212<TAB
>021:22222<TAB>060:971107131212<TAB>021:333<TAB
>900:300<TAB><ETX>
```

Enquire message status (04)

This operation is used to request a status report for a previously submitted short message.
An *enquire message status* operation can be performed independently of the status report request parameter used in the submit message operation.
No multiple enquiries are allowed in one *enquire message status* message packet, so SMEs have to request each status report separately.

Nbr	Enquire message status parameters:	Presence
021	destination address	M
060	service centre time stamp	M

The information in the *enquire message status* response is the same as in the message status delivery.

Positive response parameters:

021	destination address
060	service centre time stamp
061	status code
063	discharge time

Negative response parameters:

900	error code
901	error text (optional)

```
Enquire Message Status --->
<STX>04:003<TAB>021:DestAddr<TAB>060:ServiceCentreTimeStamp
<TAB>cs<ETX>

<--- Positive response
<STX>54:003<TAB>021:DestAddr<TAB>060:ServiceCentreTimeStamp
<TAB>061:StatusCode<TAB>063:DischargeTime<TAB>cs<ETX>
<--- Negative response
<STX>54:003<TAB>900:ErrorCode<TAB>checksum<ETX>
```

Delivery request (05)

This operation is used by the Client (SME) to retrieve a Short Message sent to the client. The usage of this operation depends on the type of SME.
The querying type of SME must always poll for messages using this Delivery Request operation.
For the receiving type of SME this operation is optional as normally messages are delivered immediately to the SME using the Deliver message operation (020). This operation can still be useful for querying the count of messages waiting for the SME.
This operation can not be used by send-only applications.
If the response to the Delivery Request with mode 1 or 2 is positive, one or more *deliver short message* operations will follow until all the messages are delivered.

Nbr:	Delivery Request:	Presence
068	mode *	O

Mode can have one of the following values (default value = 1):
0 = Number of messages waiting.
Response contains the number of messages waiting to be retrieved, no actual short messages are transferred after this request.
1= Deliver one short message.
The oldest short message is delivered.
2 = Deliver all short messages (receiving SMEs only!)
All the short messages are delivered

Positive response parameters:

066	number of messages waiting*

* The positive response contains the parameter only if the mode parameter in the request was used with value zero.

Negative response parameters:

900	error code
901	error text (optional)

Flow of operations:

SME SMSC

----> <----

DeliverRequest

DeliverRequest Rsp
DeliverMessage

DeliverMessageRsp
DeliverRequest(Mode=1)

DeliverRequest Rsp
DeliverMessage

DeliverMessageRsp
DeliverRequest(Mode=2)

DeliverRequest Rsp
DeliverMessage

DeliverMessageRsp

DeliverMessage

DeliverMessageRsp

. . .

Examples

```
A querying application using the Delivery Request for
retrieving messages

Delivery Request ----->
<STX>05:007<TAB><ETX>

<--- Positive response
<STX>55:007<TAB><chksum><ETX>

<----- Deliver message
<STX>20:002<TAB>021:DestAddr<TAB>060:ServiceCentreTimeS
tamp <TAB>023:OrigAddr<TAB>033:Data<TAB><chksum><ETX>

 Deliver message response ----->
<STX>70:002<TAB><ETX>

Delivery Request ----->
<STX>05:009<TAB><ETX>

<--- Positive response
<STX>55:009<TAB><chksum><ETX>

<----- Deliver message
<STX>20:004<TAB>021:DestAddr<TAB>060:ServiceCentreTimeS
tamp <TAB>023:OrigAddr<TAB>033:Data<TAB><chksum><ETX>

 Deliver message response ----->
<STX>70:004<TAB>chksum<ETX>
```

```
A querying or receiving application asks for the number of messages
waiting:

Delivery Request (mode=0) --->
<STX>05:011<TAB>068:0<TAB><ETX>

<--- Positive response
<STX>55:011<TAB>066:26<TAB><chksum><ETX>
```

Cancel message (06)

This operation is used to cancel an earlier sent short message. It is possible to cancel more than one message with one operation.
If the short message has already been delivered to its destination, it cannot be cancelled. Cancelling may be disabled in the submit operation. Disabling is useful for example in cases where there are such messages to a certain destination that should not be cancelled, but the cancellation is made according to the destination address.

Nbr:	Cancel message parameters:	Presence
060	service centre time stamp	O

021	destination address	O
059	cancel mode	M

The mode parameter can have three different values:

0: cancel all messages with the same destination address. This is only valid for MT messages.
1: cancel all messages sent by this subscriber. This is only valid for MT messages.
2: cancel a message where the destination address and the time stamp match with an earlier submitted SM.

In mode 0, the destination address is mandatory. In mode 1, only the cancel mode parameter is needed, and in mode 2, all three parameters are mandatory.

Cancel message response parameters:
Positive response contains no parameters.
Negative response parameters:

900	error code
901	error text (optional)

A cancel message response is positive even if the message cannot be cancelled. A positive response only means that the cancel request was received correctly by the SMSC.

Deliver message (20)

This operation is used by the SMSC to deliver a short message to a SME. Only receiving-type SMEs will get this message.

Nbr:	Deliver message parameters:	Presence
021	destination address	M
023	originator address	M
060	service centre time stamp	M
032	user data header	O
033	user data	O
034	user data binary	O
052	protocol identifier	O
030	data coding scheme	O

The *User data header* parameter is present only if the message contains a user data header. Either the user data (033) or user data binary (034) is used, depending on the value of the data coding scheme(030).
If the data coding scheme indicates that the data is using the default GSM character set, the 'normal' user data parameter is used. For UCS2 or other binary data, the user data binary (034) parameter is used.
Other optional parameters are normally present unless they are suppressed via the user interface profile.
Positive Response parameters:
Response contains no parameters.

Deliver status report (23)

This operation is used by the SMSC to send a status report describing the delivery status of a previously submitted short message. The generation of status reports for a particular short message is requested in the submit operation.
The delivery of status reports to the SME depends also on the type of SME. Querying SMEs always have to request the delivery of a status report using the operation Enquire Message Status (04)). Receiving SMEs will receive the status report whenever it is available.
After a successful delivery of the status report, the status report will be removed from the SMSC if it describes the final status of the message.
The information in the *deliver status report* message is the same as in the *enquire message status* response.

Nbr:	Deliver status report parameters:	Presence
021	destination address	M
060	service centre time stamp	M
061	status code	M
063	discharge time	M

Positive Response Parameters:
Response does not contain any parameters.

Example:

```
<---  Deliver Status Report
<STX>23:012<TAB>021:destAddr<TAB>060:scts
<TAB>061:statusCode<TAB>063:dis_time<TAB><chksum><ETX>

Response  --->
<STX>73:012<TAB><chksum><ETX>
```

Set (08)

This operation is used by the SME to change those parameters of the interface profile that it is allowed to change.
At the moment, only the password can be changed by SMEs.

Nbr:	Set parameters:
011	password

Positive Response Parameters:
Response does not contain any parameters.

Negative Response Parameters:

900	error code
901	error text (optional)

```
Set password -->
<STX>08:009<TAB>011:newPassword<TAB><chksum><ETX>

Response <---
<STX>58:009<TAB><chksum><ETX>
```

Get (09)

This operation is used by the SME to retrieve specific parameters from the SMSC.
In the Get operation, the parameter 500 is used in all get operations and the value of the parameter 500 is the number of the parameter (or information) the SME is requesting.

Nbr:	Get parameters:
500	Get parameter

Currently the only supported value for get parameters is 501 which returns the time of the SMSC.

Positive Response Parameters:

501	Time of the SMSC

Positive response contains the value parameter (or information) requested and the real value of that parameter. Negative response will follow if the parameter value is not available.

Negative response parameters:

900	error code
901	error text (optional)

```
Get --->
<STX>09:009<TAB>500:501<TAB><checksum><ETX>

<----- response
<STX>59:009<TAB>501:970211194512<TAB><checksum><ETX>
```

Alive (40)

This operation can be used by both the SME and the SMSC to check whether the link between the SME and the SMSC is still alive. The receiving entity sends an acknowledgement back to the originator if the alive operation is received correctly.

Operation parameters:

Operation does not need any parameters.

Positive Response parameters:

No parameters.

An example of an SME originated Alive request

```
Alive request -------->
<STX>40:009<TAB><Chksum><ETX>

<------- response
<STX>90:009<TAB><Chksum><ETX>
```

An example of an SMSC originated Alive request

```
<------ Alive request
<STX>40:022<TAB><Chksum><ETX>

Response ------->
<STX>90:022<TAB><Chksum><ETX>
```

Additional operations

This section list the response packages used in certain error situations as described in the previous chapter.

General error response (98)

This operation is used by the SMSC to respond to illegal or unsupported operations sent by the SME.

Operation parameters:

900	error code
901	error text (optional)

Example:

```
Request  ---->
<STX>07:007<TAB><checksum><ETX>

<---- response
<STX>98:007<TAB>900:ErrorCode<TAB><checksum><ETX>
```

Nack (99)

The *nack* message is used to reject an operation due to an incorrect checksum or an incorrect sequence number. A *nack* causes a re-transmission of the message. The packet sequence number in the *nack* message is always the expected sequence number.

Operation parameters:

No parameters.

An example of a corrupted message retransmission:

```
SME sends ------->
<STX>03:007<TAB>021:daddr<TAB>033:data<TAB>3A<ETX>

SMSC Receives
<STX>03:007<TAB>021:daddr<TAB>033:XXXX<TAB>3A<ETX>

<----- response with the nack
<STX>99:007<TAB><checksum><ETX>

SME re-transmits ------>
<STX>03:007<TAB>021:daddr<TAB>033:data<TAB>3A<ETX>

<------- SMSC response
<STX>53:007<TAB>021:daddr<TAB>060:scts<TAB><CS><ETX>
```

An example of wrong packet number:

The SMSC is expecting a packet number 007, but the client sends something else, after the nack the client recovers and the normal operation continues.

```
SME sends unexpected packet number -------->
<STX>03:001<TAB>021:daddr<TAB>033:Data<TAB><CS><ETX>

<------- SMSC responds with nack
<STX>99:009<TAB><CS><ETX>

SME recovers ---->
<STX>03:009<TAB>021:daddr<TAB>033:Data<TAB><CS><ETX>

<------- SMSC responds
<STX>53:009<TAB>021:daddr<TAB>060:60:scts<TAB><CS><ETX>
```

CIMD parameters

This chapter will explain all parameters, their maximum length in CIMD2 messages, their type and their value range.

Parameter types

The type of a parameter selects the allowed values of characters in the CIMD2 messages.

Integer (int)

This is the most common type of parameter. The allowed values are the ASCII representation of the digits 0 to 9. This means that only decimal values from 48 to 57 are allowed.
Examples:

```
064:3<tab>050:167<tab>030:0<tab>055:1<tab>056:63
```

Address (addr)

This type is used for passing GSM addresses to and from the SMSC. This field can contain digits like the integer field, but also some extra characters. The actual characters accepted depend on the configuration of the address conversion done in the SMSC. The ASCII representations of the following characters are currently allowed: '-' , '+'. This means that besides the values 48 to 57 (digits '0' to '9') also the values 45 ('-') and 43 ('+') are allowed.
Examples:

```
021:+35812345678<tab>023:13131
```

Hexadecimal (hex)

This type is used for passing binary data to and from the SMSC. This field can contain the ASCII representation of the digits 0 to 9 and the ascii representation of the characters 'A' to 'F' and 'a' to 'f'. This means the following decimal values are allowed: 48 to 57 (0 to 9); 65 to 70 ('A' to 'F') and 97 to 102 ('a' to 'f') in the CIMD2 parameters of this type.
Examples of sending six bytes of binary data:

```
034:0500032a0301<tab>
```

User Data (ud)

This type is used to exchange the user data between the SMSC and a SME. The SME can use most of the printable ASCII character set, and can combine them to select the characters of the default GSM character set. The exact configuration of this conversion is selected by the user profile. The default character conversion table is printed in appendix a.
Below is an example showing the first few characters of the default GSM character set as can be entered using the default character conversion over and 8-bit wide link using a system with the iso-latin character set.

```
033:@£$¥èéùìòÇ
which is from the isolatin character set.
48,51,51,58,64,163,36,165,232,233,249,236,242,199
```

For a 7-bit link, only values under 128 can be used. The following examples show how to enter the same data as before from a system with a different character set.:

```
033:_a0_L-$_Y-_e`_e'_u`_i`_o`_C,
```

String (str)

This type is used to exchange general printable characters from the ASCII character table. Parameters of this type can have the decimal values from 32 (space) to 126 ('~').
Example:

```
010:MyUserId<tab>011:my()<pass;+\wo%=rd<tab>
```

The parameters

Name	Id	Max-Length	Type	Values	Description

Name	Id	Max-Length	Type	Values	Description
User Identity	010	32	String		Identity used to login. Max. 32 characters.
Password	011	32	String		Password used to login. Max. 32 characters.
Destination Address	021	20	Address		Destination address in the GSM network. The prefix '+' indicates address type 145 which means international address.
Originating Address	023	20	Address		Originating address. This value is appended to the prefix in the ASE subscriber database. The total length can be at most 20 characters.
Data Coding Scheme	030	3	Integer	0-255	As defined in GSM 03.38. The value 0 indicates the default character set.
User Data Header	032	280	Hexa-decimal		This is an optional part of the user data of the message transferred. The length of the header and the user data binary fields can be at most 140 bytes. The length of the user data header combined with the (converted) user data can also be at most 140 bytes.
User Data	033	480	User Data		This parameter can be used to transfer the user data of the short message using the default character set. Special 3-character sequences can be used to select for example some Greek characters. The maximum length of the User Data in the short message itself can be at most 160 septets. This corresponds to 140 octets or bytes.
User Data Binary	034	280	Hexa-decimal		This field can be used to transfer the user data of the short message. As it uses the simple hexadecimal coding any bit pattern can be sent to another SME. By selecting an appropriate value for the data coding scheme (030) binary data or UCS2 user data can be transfered.
Validity Period Relative	050	3	Integer	-1 to 255	The length of the validity period of the short message, counted from the time the message is received by the SMSC.-The value is converted to a time period according to the GSM 03.40 specification. This parameter can not be combined with the Validity Period Absolute (051).
Validity Period Absolute	051	12	Integer		The absolute time of the termination of the validity period of the short message; value

Name	Id	Max-Length	Type	Values	Description
					consists of year, month, day, hour, minute, second in the format “yymmddhhmmss” This parameter can not be combined with the validity period relative (050)
Protocol Identifier	052	3	Integer	0 to 255	The Protocol Identifier (PID) can be used to control GSM phase 2 features, such as "replace message". Setting the PID to 65 (decimal) sets the message type to "Replace Message Type 1", adding 1..6 gives the replace message types 2 to 7, as defined in GSM 03.40. This parameter can also select the routing to applications in stead of GSM mobile stations as defined in the SMSC routing table.
First Delivery Time Relative	053	max 3	Integer	-1 to 255	This parameter selects the time period waited before the first delivery attempt of the short message will be made. The value is converted like the validity period relative.
First Delivery Time Absolute	054	12	Integer		Time for the first delivery attempt of the short message. Representation in format “yymmddhhmmss”.
Reply Path	055	1	Integer	0 or 1	This parameter selects whether the reply path feature is enabled (1) or not (0)
Status Report Request	056	2	Integer		Defines in what cases the status report shall be returned. Value of the parameter is the sum of the cases where automatic status report shall be produced. e.g. Value 62 means that status report is created for all events except temporary error. Possible values: 1 temporary error 2 validity period expired 4 delivery failed 8 delivery successful 16 message cancelled 32 message deleted by the operator
Cancel Enabled	058	1	Integer	0 or 1	This parameter selects whether the submitted message can later be cancelled (1) or not (0).
Cancel Mode	059	1	Integer	0, 1, 2	Mode used in cancelling messages.
Service Centre Time Stamp	060	12	Interger		Time of message arrival at the Service Centre. In format “yymmddhhmmss”.
Status Code	061	2	Integer		Status of the short message delivery. Passed to the SME in the inquire message

Name	Id	Max-Length	Type	Values	Description
					status response and deliver status report messages. Value as follows: 0 no status available 1 in process 2 validity period expired 3 delivery failed 4 delivery successful 5 no response 6 last no response 7 message cancelled 8 message deleted 9 message deleted by cancel
Discharge Time	063	12	Integer		Time of the last delivery attempt in format “yymmddhhmmss”.
Tariff Class	064	2	Integer	0 to 99	Defines the tariff class of the message.
Service Description	065	1	Integer	0 to 9	Defines the service description of the message which can be used for billing.
Message Count	066	3	Integer	0 to 999	Number of messages waiting to be retrieved.
Priority	067	1	Integer	1 to 9	Priority of the message. Lower value means higher priority.
Delivery Request Mode	068	1	Integer	0, 1, 2	Selects the Mode in the Delivery Request operation
Get parameter	500	3	Integer	501-999	The value is the parameter number or information item that the SME wants to retrieve. Currently only the value 501 is supported.
SMSC Time	501	12	Integer		The local time of the SMSC in the format “yymmddhhmmss” for year,month,day,hours,minutes and seconds.
Error Code	900	2	Integer	0 to 99	The identifier of the error. The error codes are listed in the next section.
Error Text	901	64	String		Text describing the error. These merely help so that the client application developer does not have to look up the error coded.

Error codes

The table below lists the error codes returned together with the error texts.

Error Code	Error Text
1	Unexpected operation
2	Syntax error
3	Unsupported parameter error
4	Connection to SMSC lost
5	No response from SMSC
6	General system error
7	Cannot find information
8	Parameter formatting error
9	Requested operation failed
	LOGIN errorcodes
100	Invalid login
101	Incorrect access type
102	Too many users with this login id
103	Login refused by SMSC
	SUBMIT MESSAGE errorcodes
300	Incorrect destination address
301	Incorrect number of destination addresses
302	Syntax error in user data parameter
303	Incorrect bin/head/normal user data parameter combination
304	Incorrect dcs parameter usage
305	Incorrect validity period parameters usage
306	Incorrect originator address usage
307	Incorrect pid parameter usage
308	Incorrect first delivery parameter usage
309	Incorrect reply path usage
310	Incorrect status report request parameter usage
311	Incorrect cancel enabled parameter usage
312	Incorrect priority parameter usage

Error Code	Error Text
313	Incorrect tariff class parameter usage
314	Incorrect service description parameter usage
	ENQUIRE MESSAGE STATUS errorcodes
400	Incorrect address parameter usage
401	Incorrect scts parameter usage
	DELIVERY REQUEST errorcodes
500	Incorrect scts parameter usage
501	Incorrect mode parameter usage
502	Incorrect parameter combination
	CANCEL MESSAGE errorcodes
600	Incorrect scts parameter usage
601	Incorrect address parameter usage
602	Incorrect mode parameter usage
603	Incorrect parameter combination
	SET errorcodes
800	Changing password failed
801	Changing password not allowed
	GET errorcodes
900	Unsupported item requested

Default user data character conversion

The table below shows the default character conversion for the user data.
The three columns on the left show the value, symbol and name of the character of the 7-bit default GSM character set as defined in the GSM 3.38.
The fourth column shows the possible character of the iso-latin character set and in brackets the decimal value. Computers using another character table can also send the value from this column even though it will look like a different character on their system. The fifth column shows the combination of iso-latin characters which result in the intended GSM character. The characters used in this column also match the corresponding character of the common ascii table.

GSM Character			Iso-Latin	Special Combi
0	@	COMMERCIAL AT	@ (64)	_Oa

GSM Character			Iso-Latin	Special Combi
1	£	POUNDS STERLING		_L-
2	$	DOLLAR SIGN	$ (36)	
3	¥	YEN		_Y-
4	è	e WITH GRAVE ACCENT		_e`
5	é	e WITH ACUTE ACCENT		_e´
6	ù	u WITH GRAVE ACCENT		_u`
7	ì	i WITH GRAVE ACCENT		_i`
8	ò	o WITH GRAVE ACCENT		_o`
9	Ç	C WITH CEDILLA		_C,
10		LINE FEED	(10)	
11	Ø	O WITH SLASH		_O/
12	ø	o WITH SLASH		_o/
13		CARRIAGE RETURN	(13)	
14	Å	A WITH RING	] (93)	_A*
15	å	a WITH RING	} (125)	_a*
16	D	GREEK ALPHABET DELTA		_gd
17	_	UNDERSCORE		_--
18	F	GREEK ALPHABET PHI		_gf
19	G	GREEK ALPHABET GAMMA		_gg
20	L	GREEK ALPHABET LAMBDA		_gl
21	W	GREEK ALPHABET OMEGA		_go
22	P	GREEK ALPHABET PI		_gp
23	Y	GREEK ALPHABET PSI		_gi
24	S	GREEK ALPHABET SIGMA		_gs
25	Q	GREEK ALPHABET THETA		_gt
26	X	GREEK ALPHABET XI		_gx
27		RESERVED		_XX
28	Æ	AE DIPHTHONG		_AE
29	æ	ae DIPHTHONG		_ae
30	ß	GERMAN DOUBLE-S		_ss
31	É	E WITH ACUTE ACCENT		_E´
32		SPACE	(32)	
33	!	EXCLAMATION MARK	! (33)	
34	"	QUOTATION MARK	" (34)	_qq
35	#	NUMBER SIGN	# (35)	

GSM Character			Iso-Latin	Special Combi
36	¤	CURRENCY SYMBOL		_ox
37	%	PERCENT SIGN	% (37)	
38	&	AMPERSAND	& (38)	
39	'	APOSTROPHE	' (39)	
40	(	LEFT PARENTHESIS	((40)	
41	)	RIGHT PARENTHESIS	) (41)	
42	*	ASTERISK	* (42)	
43	+	PLUS SIGN	+ (43)	
44	,	COMMA	, (44)	
45	-	HYPHEN	- (45)	
46	.	FULL STOP (PERIOD)	. (46)	
47	/	SOLIDUS (SLASH)	/ (47)	
48	0	DIGIT ZERO	0 (48)	
49	1	DIGIT ONE	1 (49)	
50	2	DIGIT TWO	2 (50)	
51	3	DIGIT THREE	3 (51)	
52	4	DIGIT FOUR	4 (52)	
53	5	DIGIT FIVE	5 (53)	
54	6	DIGIT SIX	6 (54)	
55	6	DIGIT SEVEN	7 (55)	
56	7	DIGIT EIGHT	8 (56)	
57	8	DIGIT NINE	9 (57)	
58	9	COLON	: (58)	
59	;	SEMICOLON	; (59)	
60	<	LESS-THAN SIGN	< (60)	
61	=	EQUALS SIGN	= (61)	
62	>	GREATER-THAN SIGN	> (62)	
63	?	QUESTION MARK	? (63)	
64	¡	INVERTED !		_!!
65	A	CAPITAL LETTER A	A (65)	
66	B	CAPITAL LETTER B	B (66)	
67	C	CAPITAL LETTER C	C (67)	
68	D	CAPITAL LETTER D	D (68)	
69	E	CAPITAL LETTER E	E (69)	
70	F	CAPITAL LETTER F	F (70)	

GSM Character			Iso-Latin	Special Combi
71	G	CAPITAL LETTER G	G (71)	
72	H	CAPITAL LETTER H	H (72)	
73	I	CAPITAL LETTER I	I (73)	
74	J	CAPITAL LETTER J	J (74)	
75	K	CAPITAL LETTER K	K (75)	
76	L	CAPITAL LETTER L	L (76)	
77	M	CAPITAL LETTER M	M (77)	
78	N	CAPITAL LETTER N	N (78)	
79	O	CAPITAL LETTER O	O (79)	
80	P	CAPITAL LETTER P	P (80)	
81	Q	CAPITAL LETTER Q	Q (81)	
82	R	CAPITAL LETTER R	R (82)	
83	S	CAPITAL LETTER S	S (83)	
84	T	CAPITAL LETTER T	T (84)	
85	U	CAPITAL LETTER U	U (85)	
86	V	CAPITAL LETTER V	V (86)	
87	W	CAPITAL LETTER W	W (87)	
88	X	CAPITAL LETTER X	X (88)	
89	Y	CAPITAL LETTER Y	Y (89)	
90	Z	CAPITAL LETTER Z	Z (90)	
91	Ä	A WITH DIERESIS	[(91)	_A"
92	Ö	O WITH DIERESIS	\ (92)	_O"
93	Ñ	N WITH TILDE		_N~
94	Ü	U WITH DIERESIS		_U"
95	§	SECTION MARK		_so
96	¿	INVERTED ?		_??
97	a	SMALL LETTER a	a (97)	
98	b	SMALL LETTER b	b (98)	
99	c	SMALL LETTER c	c (99)	
100	d	SMALL LETTER d	d (100)	
101	e	SMALL LETTER e	e (101)	
102	f	SMALL LETTER f	f (102)	
103	g	SMALL LETTER g	g (103)	
104	h	SMALL LETTER h	h (104)	
105	i	SMALL LETTER i	I (105)	

GSM Character			Iso-Latin	Special Combi
106	j	SMALL LETTER j	j (106)	
107	k	SMALL LETTER k	k (107)	
108	l	SMALL LETTER l	l (108)	
109	m	SMALL LETTER m	m (109)	
110	n	SMALL LETTER n	n (110)	
111	o	SMALL LETTER o	o (111)	
112	p	SMALL LETTER p	p (112)	
113	q	SMALL LETTER q	q (113)	
114	r	SMALL LETTER r	r (114)	
115	s	SMALL LETTER s	s (115)	
116	t	SMALL LETTER t	t (116)	
117	u	SMALL LETTER u	u (117)	
118	v	SMALL LETTER v	v (118)	
119	w	SMALL LETTER w	w (119)	
120	x	SMALL LETTER x	x (120)	
121	y	SMALL LETTER y	y (121)	
122	z	SMALL LETTER z	z (122)	
123	ä	a WITH DIERESIS	{ (123)	_a"
124	ö	o WITH DIERESIS	| (124)	_o"
125	ñ	n WITH TILDE		_n~
126	ü	u WITH DIERESIS		_u"
127	à	a WITH GRAVE ACCENT		_a`

Example sessions

Example session for a send-only SME

The example session given here contains the operations login, submit message and logout. Note that in the submit operation example, only parameters *destination address*, *validity period* and *user data* are given by the SME. Note that packets are split at special characters like <TAB> for printing purposes only.

SME <--> SMSC
Connect <-->
Login -->

<STX>01:001<TAB>010:MyUserId<TAB>011:MySecretPassWOrD<TAB><CS><ETX>

Login response <--

<STX>51:001<TAB><CS><ETX>

Submit -->

<STX>03:003<TAB>021:1234567890<TAB>050:167<TAB>033:Hi there !<TAB><CS><ETX>

Submit response (positive) <--

<STX>53:003<TAB>021:1234567890<TAB>060:971111131245<TAB><CS><ETX>

Submit -->

<STX>03:005<TAB>050:167<TAB>033:Hi there ! How are you doing ?<TAB><CS><ETX>

Submit response (negative) //destination was missing!! <--

<STX>53:005<TAB>900:301<TAB><CS><ETX>

Enquiry message status-->

<STX>04:007<TAB>060:971111131245<TAB>021:1234567890<TAB><CS><ETX>

Enquiry message status response <--

<STX>54:007<TAB>021:1234567890<TAB>060:971111131245<TAB>061:0<TAB->063:971111131245<TAB><CS><ETX>

Disconnect <-->

Example session for querying SME

SME <--> **SMSC**

Connect <-->

Login -->

<STX>01:001<TAB>010:MyUserId<TAB>011:MySecretPassWOrD<TAB><CS><ETX>

Login response <--

<STX>51:001<TAB><CS><ETX>

Submit -->

<STX>03:003<TAB>021:1234567890<TAB>050:167<TAB>033:Hi there !<TAB><CS><ETX>

Submit response (positive) <--

<STX>53:003<TAB>021:1234567890 <TAB>060:971111131245<TAB><CS><ETX>

Delivery Request Message -->

<STX>05:005<TAB>068:1<TAB><CS><ETX>

Delivery Request Response (positive) <--

<STX>55:005<TAB><CS><ETX>

Deliver Message <--

<STX>20:000<TAB>021:1234567890<TAB>023:987654321<TAB>060:971111121245<TAB->033:Call me.<TAB><CS><ETX>

Deliver Message Response -->

```
<STX>70:000<TAB><CS><ETX>
```

Disconnect <**--**>

Example session for receiving SME

SME <--> **SMSC**
Connect <-->
Login -->

```
<STX>01:001<TAB>010:MyUserId<TAB>011:MySecretPassWOrD<TAB><CS><ETX>
```

Login response <--

```
<STX>51:001<TAB><CS><ETX>
```

Submit -->

```
<STX>03:003<TAB>021:1234567890<TAB>050:167<TAB>033:Hi
  there!<TAB><CS><ETX>
```

Deliver Message //delivery before submit response received <--

```
<STX>20:000<TAB>021:9876<TAB>023:1234567890<TAB>060:971111080045<TAB-
  >033:Call me!<TAB>052:0<TAB>030:0<TAB><CS><ETX>
```

Deliver Message Response -->

```
<STX>70:000<TAB><CS><ETX>
```

Submit response (positive) <--

```
<STX>53:003<TAB>021:1234567890<TAB>060:971111131245<TAB><CS><ETX>
```

Deliver Message <--

```
<STX>20:002<TAB>021:98765<TAB>023:1234567890<TAB>060:971111121245<TAB
  >033:Why dont you call me<TAB>052:0<TAB>030:0<TAB><CS><ETX>
```

Deliver Message Response -->

```
<STX>70:002<TAB><CS><ETX>
```

Deliver StatusReport <--

```
<STX>23:005<TAB>021:1234567890<TAB>060:971111131245<TAB>061:3<TAB
  >063:971111131354<TAB><CS><ETX>
```

Deliver StatusReport Response -->

```
<STX>73:005<TAB><CS><ETX>
```

Disconnect <**--**>

Index

A

C

D

E

L

M

N

O

P

R

S