

**Najah National University**  
Computer Engineering Department

# **SBOX Version 1.0**

## **Programmers Manual**

Supervisors

Dr.Raed Al Qadi & Dr. Loai Malhees

SBox Project By

Wafa N.Adham & Manar L.Qamhieh



07

## What is SBox?

SBox [Student Box], is a unique package that enables the students to interact with their instructors any time anywhere, this system is designed for educational institutes in general and for Najah National University in specific .

## Why SBox?

As the educational interaction is vital these days SBox comes with new creative ideas on how this interaction can be maximized allowing students to reach their accounts with the simplest , easiest and cost-effective ways , provide them with the updated information they need to know about their courses and university news.

Personal attendance was the traditional way of interaction between student and the organization, but this is not practical all.

The internet revolution changed the world and offered an amazing solutions to such problems, it came as a reasonable solution to this situation, and it gained a great reputation, and became the backbone of any educational system, but it is restricted with the internet connectivity, which is not available everywhere especially in far palestinian villages.

This is the era of the mobile devices , micro-computers and wireless services, where these devices are available to huge section of the palestinian community , the existence of many communication companies providing us with a large number of mobile services at low costs made us think more about exploiting these facilities in brand new product that serves most palestinian students .also because the cost o f these devices is reasonable and available everywhere most students have at least one mobile device , these reasons and much others leaded us to the creation the[ SBox-Mobile version].

Since there is a large number of different mobile devices that have different capabilities and features SBox offered multiple mobile services , separated into four parts as follows:

- 1- **SBox Web Application:** this represents the backbone of our project enables the students to have their own accounts and interact from their regular pc's Browsers such as IE ,or Mozilla firefox.
- 2- **SBox-Mobile Edition:** this is the java mobile application, enables mobiles who have built in java virtual machine to connect to their accounts using tiny application can that can be downloaded easily from the SBox web application.
- 3- **SBox-WAP:** This was made especially for devices who doesn't have built-in java virtual machine and does have mobile micro-browser.
- 4- **SBox-SMS:** this a two way sms service enables students to send an sms to specific number and the they get response from the SBox system with the service they asked after authentication of course.

## SBox Java Application:

J2ME: Java 2 Platform, Micro Edition (J2ME) is the second revolution in Java's short history. Where small programs can be downloaded and run on demand.

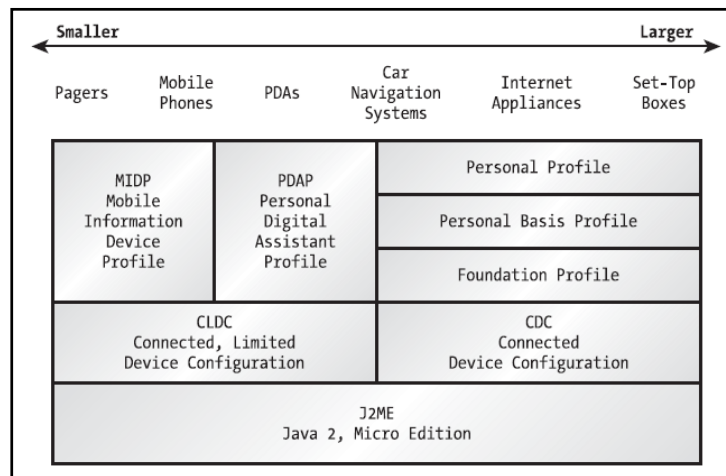
J2ME isn't a specific piece of software or specification. All it means is Java for small devices.

Small devices range in size from pagers, mobile phones, and personal digital assistants (PDAs).

J2ME is divided into configurations, profiles, and optional *APIs*, which provide specific information about APIs and different families of devices. A configuration is designed for a specific kind of device based on memory constraints and processor power. It specifies a Java Virtual Machine (JVM) that can be easily ported to devices supporting the configuration. It also specifies a strict subset of the Java 2 Platform, Standard Edition (J2SE) APIs that will be used on the platform, as well as additional APIs that may be necessary. Device manufacturers are responsible for porting a specific configuration to their devices.

### Mobile Information Device Profile (MIDP):

MIDP applications are piquantly called MIDlets, a continuation of the naming theme begun by applets and servlets. Writing MIDlets is relatively easy for a moderately experienced Java programmer. After all, the programming language is still Java. Furthermore, many of the fundamental APIs from `java.lang` and `java.io` are basically the same in the MIDP as they are in J2SE.



According to the MIDP 2.0 specification (JSR-118), a Mobile Information Device has the following characteristics:

- A minimum of 256KB of ROM for the MIDP implementation (this is in addition to the requirements of the CLDC).
- A minimum of 128KB of RAM for the Java runtime heap.
- A minimum of 8KB of nonvolatile writable memory for persistent data.
- A screen of at least 96×54 pixels.
- Some capacity for input, either by keypad, keyboard, or touch screens.
- Two-way network connection, possibly intermittent.

The APIs available to a MIDP application come from packages in both CLDC and MIDP.

CLDC defines a core of APIs, mostly taken from the J2SE world. These include fundamental language classes in `java.lang`, stream classes from `java.io`, and simple collections from `java.util`. CLDC also specifies a generalized network API in `javax.microedition.io`.

#### Advantages of MIDP

##### 1- Portability:

The advantage of using Java over using other tools for small device application development is portability. You could write device applications with C or C++, but the result would be specific to a single platform. An application written using the MIDP APIs will be directly portable to any MIDP device.

##### 2- Security:

Java is well known for its ability to safely run downloaded code like applets. This is a perfect fit—it's easy to imagine nifty applications dynamically downloading to your mobile phone.

#### **The user Interface:**

Many MIDP applications are built to run on many different devices without modification.

This is particularly difficult in the area of the user interface because devices have screens of all sizes, in grayscale and in color. Furthermore, devices vary widely in their input capabilities, from numeric keypads to alphabetic keyboards, soft keys, and even touch screens. The minimum screen size mandated by MIDP is 96×54 pixels, with at least one bit of color depth.<sup>1</sup> As for input, MIDP is fairly open ended: devices are expected to have some type of keyboard, or a touch screen, or possibly both.

Given the wide variety of devices that are compliant with MIDP, there are two ways to create applications that work well on all devices:

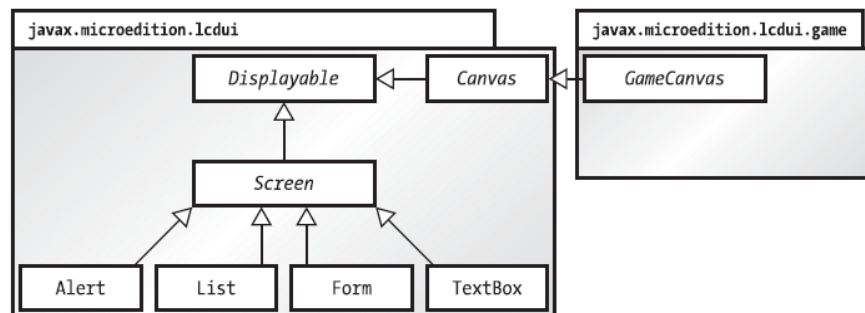
- *Abstraction*: Specify a user interface in abstract terms, relying on the MIDP implementation to create something concrete. Instead of saying something like, “Display the word ‘Next’ on the screen above the soft button,” you say, “Give me a Next command somewhere in this interface.”

- *Discovery*: The application learns about the device at runtime and tailors the user interface programmatically. You might, for example, find out how big the device’s screen was in order to scale your user interface appropriately.

The MIDP APIs support both methods. Abstraction is the preferred method because it involves less code in your application and more work by the MIDP implementation. In some cases, like games, you need to be more specific about the user interface; these types of applications will discover the capabilities of a device and attempt to tailor their behavior appropriately.

MIDP’s user interface APIs are designed so that it’s easy to mix the two techniques in the same application.

MIDP contains user interface classes in the `javax.microedition.lcdui` and `javax.microedition.lcdui.game` packages. The device’s display, as seen by the MIDlet, is represented by an instance of the `Display` class, accessed from a factory method, `getDisplay()`. `Display`’s main purpose in life is to keep track of what is currently shown.



### Gauge:

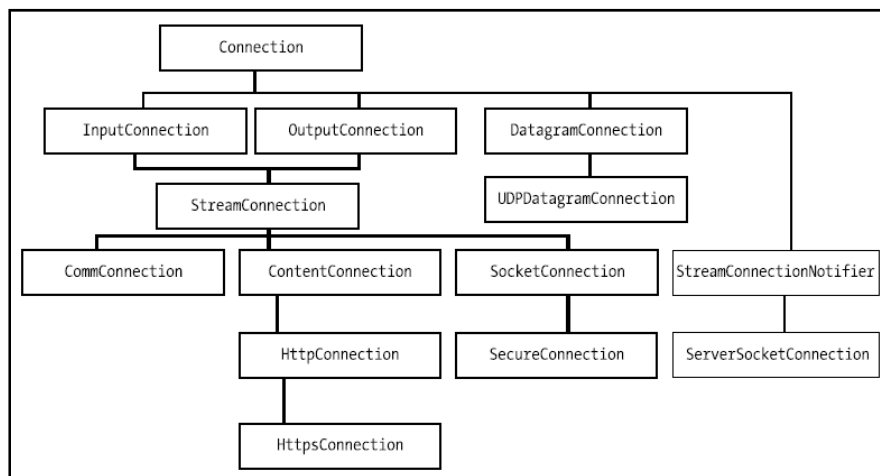
Gauge is used as a waiting screen, usually between 2 forms where data are expected to take some time to be loaded, and the gauge is much better than nothing, that because it gives the user an indicator that the application is in process and he/she has to wait a little.

### List:

Lists allow the user to select items from specified choices. and the lists are the main tool used in our application. And the list needs a `commandListener` to handle the event happens after making the choice,

### Connection to the World:

The CLDC defines an extremely flexible API for network connections, the *generic connection framework*. The core GCF is contained in the `javax.microedition.io` package and based around the `Connection` interface. Figure 10-1 details the `Connection` interface and its various child interfaces.



MIDP 2.0 requires support of Hypertext Transfer Protocol (HTTP) connections. You pass an HTTP URL to `Connector` and get back an implementation of `HttpConnection`. Mandatory support for HTTPS connections (secure HTTP) is also required by MIDP 2.0. There are also standardized connection strings for several types of connections.

Requests and responses have two parts: headers and content. If you type a URL into your browser, the browser creates an HTTP request (mostly headers) and sends it to a server. The server finds the requested file and sends it back in an HTTP response. The response headers describe things like the type of web server, the file type of the response, the length of the response, and other information. The response content is the data of the file itself.

Browsers and other HTTP clients request specific named resources from HTTP servers. In addition, clients can pass parameters to the server. Parameters are simple name and value pairs.

The client encodes parameters before they are sent to the server. Parameters are passed as name and value pairs; multiple parameters are separated by ampersands. The exact way that parameters are encoded is documented in the J2SE documentation for `java.net.URLEncoder`.

The rules are relatively simple.

1. The space character is converted to a plus (+) sign.
2. The following characters remain unchanged: lowercase letters a through z, uppercase letters A through Z, the numbers 0 through 9, the period (.), the hyphen (-), the asterisk (\*), and the underscore (\_).
3. All other characters are converted into “%xy”, where “xy” is a hexadecimal number that represents the low 8 bits of the character.

The simplest HTTP operation is GET, with a GET request; parameters are added to the end of the URL in encoded form. (Some servers have trouble with very long URLs; if you have a lot of parameters, or binary data, you may wish to pass data in the body of the HTTP request.)

POST is basically the same as GET, but parameters are handled differently. Instead of being pasted on the end of the URL, as they are with GET, the parameters are passed as the body of the request. They are encoded in the same way.

## Connection

Loading data from a server is startlingly simple, particularly if you’re performing an HTTP GET. Simply pass a URL to `Connector`’s static `open ()` method. The returned `Connection` will probably be an implementation of `HttpConnection`, but you can just treat it as an `InputConnection`. Then get the corresponding `InputStream` and read data to your heart’s content.

In code, it looks something like this:

```
String url = "http://www.sbox.com";

URLConnection ic = (URLConnection)Connector.open(url);

InputStream in = ic.openInputStream();

// Read stuff from the InputStream

ic.close();
```

Most of the methods involved can throw a `java.io.IOException` and network access is subject to security policies on the device.

## **Multithreading**

Using threads is much as you remember it from J2SE, as long as you keep things simple. Creating new threads, starting them, and using the handy `java.lang.Runnable` interface are the same as in J2SE. One important omission in CLDC 1.0 is the `interrupt ()` method, which is not present in the `java.lang.Thread` class. In CLDC 1.1, the `interrupt ()` method is available. The `pause ()`, `resume ()`, and `stop ()` methods (which are deprecated in the J2SE SDK) are also absent. Thread groups and daemon threads are not supported in CLDC/MIDP; thread naming is not supported in CLDC 1.0 but is available in CLDC 1.1.



## **SBox Web Interface:**

Our main goal - while creating SBox - is to create a nice, user-friendly web interface, to be the backbone of our system, and all the other parts are related and connected to it, and controlled through it.

### **New Features:**

The idea of the web interface for the educational institutions is an old one, and it is used nowadays, and it facilitates the communication process of these organizations. So SBox System aimed to add other mobile parts for the system. But the web interface maintained to be the backbone of all other parts, that because the web interface in general is easier to deal with, more options can be added to it easily, and it is more flexible, and widespread between the students.

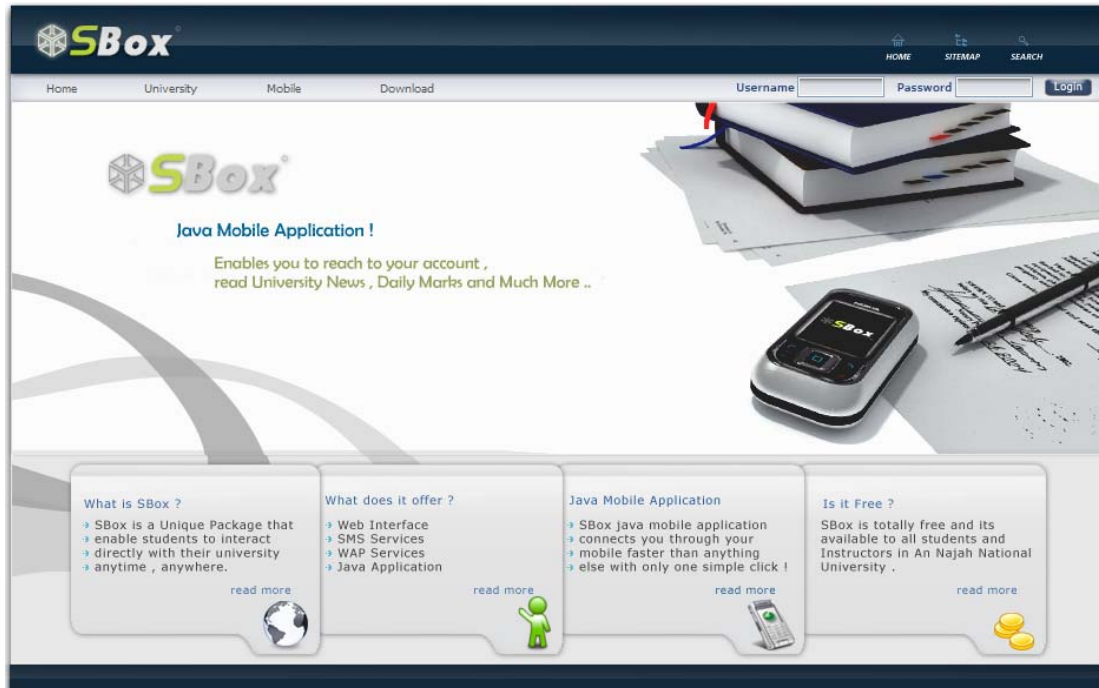
So we tried in our system to give the web interface new and important features, and they are as the following:

- 1- User Interface:
- 2- Secure Socket Layer (SSL):
- 3- Flexible Architecture:
- 4- Security and Log system:

### **User Interface:**

User interface for any web site is an important issue, and it encourages the users to keep browsing the web site and keep using it.

So we cared a lot about the interface in SBox system, and spend a lot of time so as to be as we imagined, without any using to any pre-made templates or anything else, we created the user interface from scratch and we tried to make it user friendly as much as possible.



**Figure: SBox Home Page.**

We also took care about the menus and news rollers to be in form easier for the user to use and manipulate with. And the menus are customized for each type of users.

### **Secure Socket Layer:**

It is a protocol used to provide a secure connection on the internet, for such things as web browsing, e-mail, Internet faxing, instant messaging and other data transfers.

SSL allows applications to communicate across a network in a way designed to prevent eavesdropping, tampering, and message forgery. TLS provides endpoint authentication and communications privacy over the Internet using cryptography.

So as to use SSL in our system, we configured the Apache server to support SSL protocol ,to protect the data transferred in our system that in any educational organization deals with information about the student's educational level and also there maybe a data about money and registration.

### **Website Architecture:**

SBox Architecture was designed to ensure the maximum flexibility to allow it to be updated easily, the hierarchical method was used to implement all functionalities making the system as a multiple modules, also the design pages were separated from the code pages which enables design to be updated in no time.

### **Security & Log System:**

Security is a great issue in the SBox system, since each user has to have a separate and private account, and the logging to each account must be secure each according to its privilege level (Administrator, teacher, student). Each account is protected by a username (which is the user's unique ID) and a password. And inside the account there is an option for changing the password.

And since there are other ways of accessing the account from mobile devices, we added another way of security, that each user can either enable or disable remote access to his account, and keep the web interface to be the only way of accessing.

New feature added for security was the Log System that is used in SBox. It is an option that works like an archive system and tracking system as well, from this option, the administrator can see all the accesses to the website, and keep the information about each access, including the user which accessed the site, and the date and time of access, and the IP of the computer. And the log system is provided with a filtering option, so as to facilitate the tracking and searching processes for the administrator.

The log system main functions have great importance in the management of the site, from the security perspective, it helps the administrator to monitor the accesses to the website, and it will help also to improve the web site by noticing the most accessed pages and devices, and improve and develop these pages the most, or try to fix or improve the other pages to make them more desirable.

The log system is controlled from the administrator account, that it may be activated or deactivated from there.

### **SBox Users and Account:**

SBox web interface has 3 kinds of accounts, one for the administrator, the teacher and the student. And each one of these accounts has its own options and privileges and functionality.

### Administrator's Account:

The Administrator is the one with the highest level of privileges, and he/she can control the most critical options of the educational system, like adding, editing, deleting many options like users (including teachers and student) and courses and assigning these courses to teachers and locations and time, so as to be functional to the students to register in.

And the administrator has the option of monitoring the website- as it was mentioned earlier as the Log System- , and the administrator is responsible for configuring and choosing of the SMS gateways, so as to provide a proper service of SMS messages.

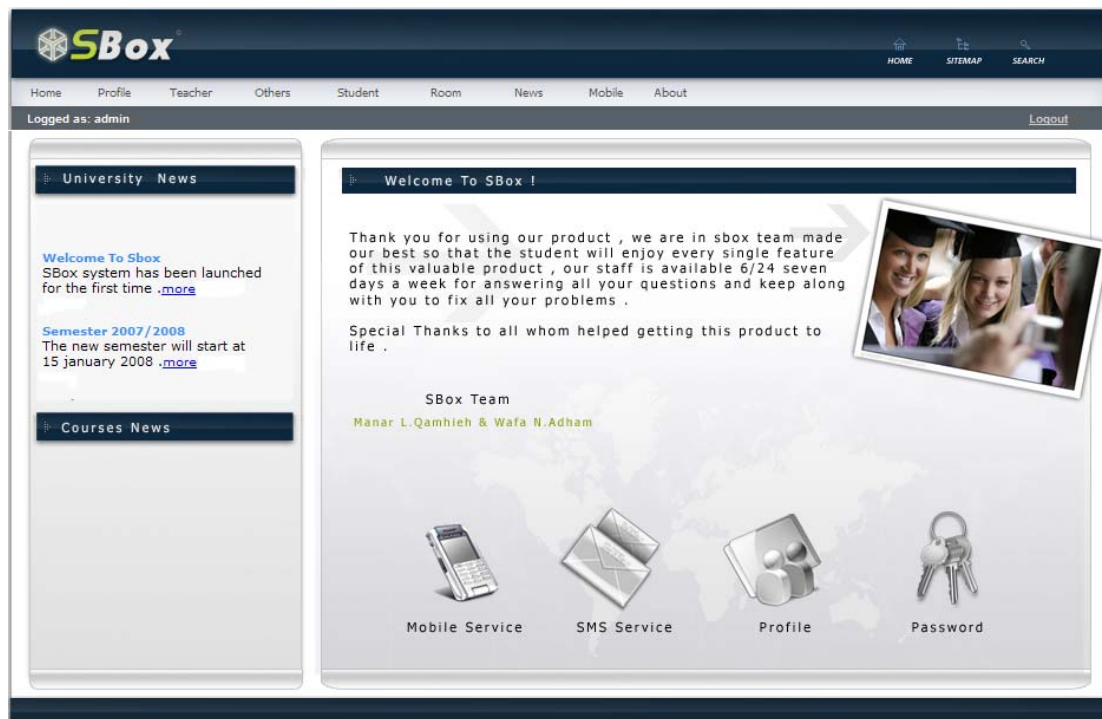
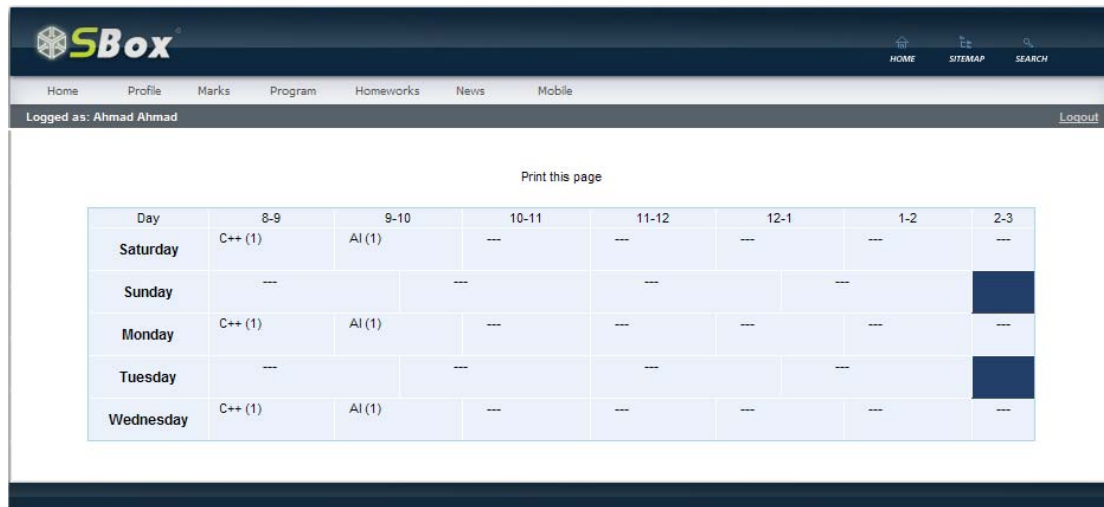


Figure: Administrator Main Page.

### Teacher's Account:

The teacher account has the options related to the teacher's work that through his/her account; the teacher can track the courses he is assigned to, and keep in contact with students by sending HWs and Marks to them through the website. And he may choose the option of sending the information through SMS.



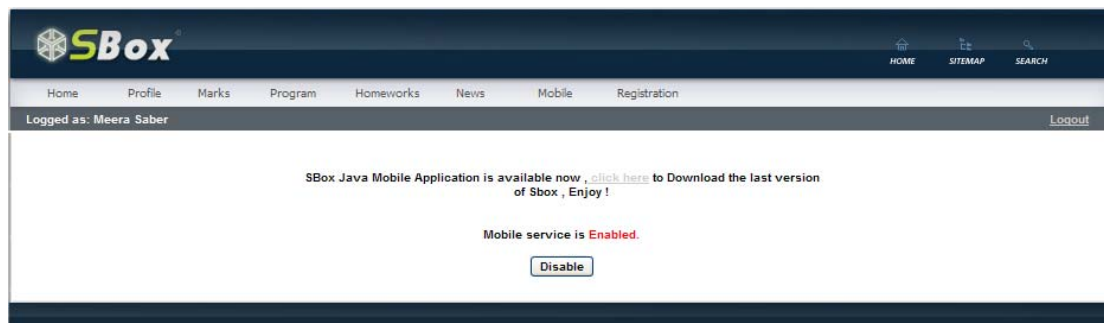
Day	8-9	9-10	10-11	11-12	12-1	1-2	2-3
Saturday	C++ (1)	AI (1)	---	---	---	---	---
Sunday	---	---	---	---	---	---	---
Monday	C++ (1)	AI (1)	---	---	---	---	---
Tuesday	---	---	---	---	---	---	---
Wednesday	C++ (1)	AI (1)	---	---	---	---	---

**Figure: Teacher's Schedule Page.**

### Student's Account:

The student's account deals with the options that most attract the student's educational life, that through his/her account he can register for new courses and see his registered courses, and keep updated by his marks, HWs and news.

And from the account he/she can control other parts of SBox system, as activating or deactivating the mobile services, which increase the security for the account.



**Figure: Student's Mobile control page.**

### User Help:

Each page contains a user help tips, which provides an instant online user help, giving the users a quick idea of how to deal with the pages and options, and a brief description about each option in the web interface, and the help tags are added to the pages that we thought they need some more description to the user.

# SBox SMS Messaging System

## Introduction

Sbox Messaging system is divided into two parts, the first part was making a local gateway that use the regular mobile phone in order to manage the Incoming Text Messages and reply according to the content of this Message, the second part was integrating SBox Web interface with a Local / Global SMS gateway Operator.

## SBox Gateway

The sBox gateway exploits the Mobile AT commands and get benefits what these valuable commands offers talking to the regular mobile phone , through the project we worked on Nokia Mobile phones for testing purposes but as most of those commands are common to most vendors then we the SBox gateway should not have any problems dealing with other mobiles that supports AT commands (SonyErricson , Motorola ,etc...) .

## Brief look on AT Commands

AT commands are instructions used to control a modem. AT is the abbreviation of ATtention. Every command line starts with "AT" or "at". That's why modem commands are called AT commands. Many of the commands that are used to control wired dial-up modems, such as ATD (Dial), ATA (Answer), ATH (Hook control) and ATO (Return to online data state), are also supported by GSM/GPRS modems and mobile phones. Besides this common AT command set, GSM/GPRS modems and mobile phones support an AT command set that is specific to the GSM technology, which includes SMS-related commands like AT+CMGS (Send SMS message), AT+CMSS (Send SMS message from storage), AT+CMGL (List SMS messages) and AT+CMGR (Read SMS messages).

Note that the starting "AT" is the prefix that informs the modem about the start of a command line. It is not part of the AT command name. For example, D is the actual AT command name in ATD and +CMGS is the actual AT command name in AT+CMGS. However, some books and web sites use them interchangeably as the name of an AT command.

Here are some of the tasks that can be done using AT commands with a GSM/GPRS modem or mobile phone:

- Get basic information about the mobile phone or GSM/GPRS modem. For example, name of manufacturer (AT+CGMI), model number (AT+CGMM), IMEI number (International Mobile Equipment Identity) (AT+CGSN) and software version (AT+CGMR).
- Get basic information about the subscriber. For example, MSISDN (AT+CNUM) and IMSI number (International Mobile Subscriber Identity) (AT+CIMI).
- Get the current status of the mobile phone or GSM/GPRS modem. For example, mobile phone activity status (AT+CPAS), mobile network registration status (AT+CREG), radio signal strength (AT+CSQ), battery charge level and battery charging status (AT+CBC).

- Establish a data connection or voice connection to a remote modem (ATD, ATA, etc).
- Send and receive fax (ATD, ATA, AT+F\*).
- Send (AT+CMGS, AT+CMSS), read (AT+CMGR, AT+CMGL), write (AT+CMGW) or delete (AT+CMGD) SMS messages and obtain notifications of newly received SMS messages (AT+CNMI).
- Read (AT+CPBR), write (AT+CPBW) or search (AT+CPBF) phonebook entries.
- Perform security-related tasks, such as opening or closing facility locks (AT+CLCK), checking whether a facility is locked (AT+CLCK) and changing passwords (AT+CPWD). (Facility lock examples: SIM lock [a password must be given to the SIM card every time the mobile phone is switched on] and PH-SIM lock [a certain SIM card is associated with the mobile phone. To use other SIM cards with the mobile phone, a password must be entered.])
- Control the presentation of result codes / error messages of AT commands. For example, you can control whether to enable certain error messages (AT+CMEE) and whether error messages should be displayed in numeric format or verbose format (AT+CMEE=1 or AT+CMEE=2).
- Get or change the configurations of the mobile phone or GSM/GPRS modem. For example, change the GSM network (AT+COPS), bearer service type (AT+CBST), radio link protocol parameters (AT+CRLP), SMS center address (AT+CSCA) and storage of SMS messages (AT+CPMS).
- Save and restore configurations of the mobile phone or GSM/GPRS modem. For example, save (AT+CSAS) and restore (AT+CRES) settings related to SMS messaging such as the SMS center address.

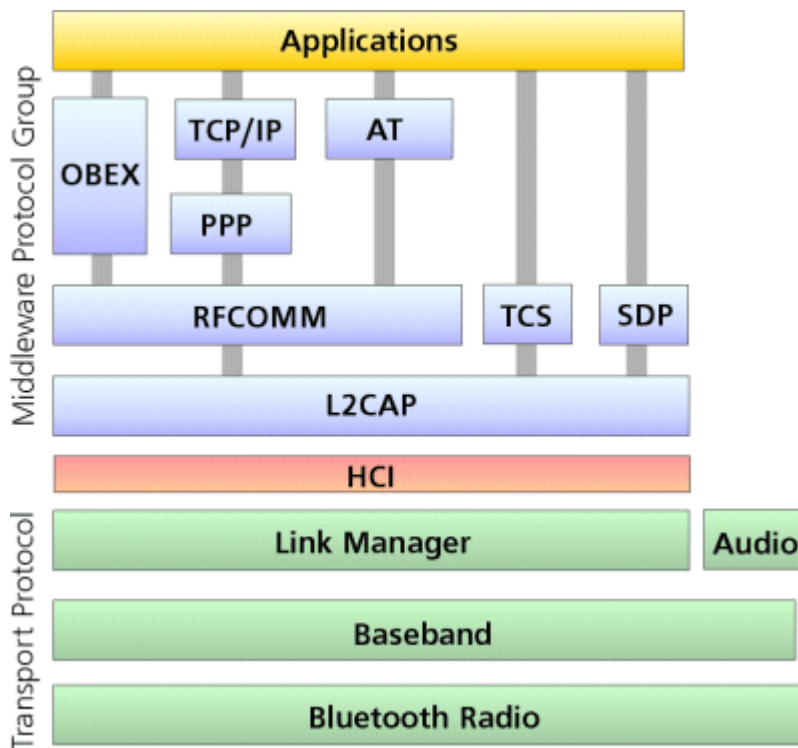
Note that mobile phone manufacturers usually do not implement all AT commands, command parameters and parameter values in their mobile phones. Also, the behavior of the implemented AT commands may be different from that defined in the standard. In general, GSM/GPRS modems designed for wireless applications have better support of AT commands than ordinary mobile phones.

In addition, some AT commands require the support of mobile network operators. For example, SMS over GPRS can be enabled on some GPRS mobile phones and GPRS modems with the +CGSMS command (command name in text: Select Service for MO SMS Messages). But if the mobile network operator does not support the transmission of SMS over GPRS, you cannot use this feature.

## SBox Implementation

For SBox gateway we used .NET C# , to speed up things we used AT Library and used Bluetooth Protocol (RFCOMM) to communicate with the GSM mobile GPRS modem , we used MySql Database to keep records of the Incoming ,Outgoing calls and SMS text messages .

The following is the Bluetooth Protocol and the Layer used to complete our task is the RFCOMM which simulates the regular serial port communication .



## SBox Gateway Requirements

The following are the hardware / software needed for SBox Gateway to work properly:

### 1. Computer Hardware:

- CPU - Pentium 233 MHz
- RAM - 128MB
- At least 10MB free hard disk space
- At least one free Serial Port or Infrared Port or USB Port or Bluetooth

**Note:** For GSM Modems or GSM Phones connected using Infrared port or USB to Serial Converter, select appropriate virtual serial port to communicate with SBox Gateway™.



## 2. GSM Modem With SIM Card

SBox Gateway™ has been designed to work with any ETSI GSM 07.05 (Version 7.0.1 Release 1998) compatible GSM modem. Any GSM mobile phone (with suitable serial/USB data cable or Infrared port or Bluetooth) that have built in modem and support AT commands can be used with SBox Gateway™. Some of the popular GSM modems with which SBox Gateway™ has been tested are as follows:

- Siemens MC35i, TC35i
- Nokia N30, 6610, 7250, 7210, 8310, 8210, N32

A valid SIM card which has access to SMS service is required with the GSM modem.

**Note:** Most Nokia phone models do not support AT command set directly. To use SBox Gateway™ with Nokia phones you may need to install Nokia Data Suite software or the appropriate modem driver driver and select the virtual serial port for communication between SBox Gateway™ and Nokia handset.

## 3. Software:

- Microsoft® .NET Framework Version 2.0
- Operating System: Windows 98/2000/XP/2003
- Any programming environment that supports .NET Class Library (e.g. Visual Basic, C#, ASP.NET etc)

*The code used to implement the AT commands and the library is included in the attached CD.*

## Implementing the Second Option (External Gateway)

For implementing the External Gateway option in Our project , we choose “Clickatell Sms gateway” as an operator .

### Clickatell Services

- Integration to all kinds of front-end and legacy systems is carried out via a range of simple, flexible Application Programming Interface (API) connections. These offer you a choice of connection options (HTTP/S, SMPP, SMTP, FTP, XML, Com Object.) which are quick and easy to implement and are suited to a wide spectrum of usage.
- The user-friendly web based interface easily allows us to manage all our API connections, monitor our campaigns, do extensive reporting, make online payments, receive invoices, add sub-users and more.
- We can choose from different message routing profiles to suit your budget and redundancy requirements.

This is what made us choose Clickatell among 10's of sms Providers , for SBox we used the Http connection to communicate with the company , sending and receiving Url's are described in the attached file and can be downloaded from the company's website [www.clickatell.com](http://www.clickatell.com) .

Clickatell http commands are made up of three segments: authentication, the basic message components (message content and recipients) and the additional message parameters. In the examples

below, we will include the authentication and basic message components. The additional message parameters will be included only where they are relevant.

Basic Command Structure:

URL Call Authentication Message Parameters

[http://api.clickatell.com/http/sendmsg?session\\_id=xxxx&to=xxxx&text=xxxx](http://api.clickatell.com/http/sendmsg?session_id=xxxx&to=xxxx&text=xxxx)

or

[http://api.clickatell.com/http/sendmsg?api\\_id=xxxx&user=xxxx&password=xxxx&to=xxxx&text=xxxx](http://api.clickatell.com/http/sendmsg?api_id=xxxx&user=xxxx&password=xxxx&to=xxxx&text=xxxx)

In order to deliver a message, the system needs to authenticate the request as coming from a valid

source. We use a number of parameters to achieve this:

\$ **api\_id**: This is issued upon addition of an HTTP sub-product to your account. A single account may have multiple API IDs associated with it.

\$ **user**: This is the username of your account.

\$ **password**: The current password you have set on your account.

The following is a sample code that we used in order to send SMS message through the provider company.

```
<?php
```

```
include ('../config.php');
echo('<style type="text/css">
.style1 {
    font-family: Geneva, Arial, Helvetica, sans-serif;
    font-size: 10px;
    font-weight: bold;
}
</style>');
```

```
echo('<span class="style1"><p align="center">');
if($sms_send_flag!="false")
{
```

```
    $q1 = "select * from settings where enabled='1'";
    $r1 = mysql_query($q1);
    $r = mysql_fetch_array($r1);
    $user = $r['name'];
    $password = $r['password'];
    $api_id = $r['client_id'];
    if ($r['carrier_name'] == "Clickatell Ltd")
    {
        $baseurl = "http://api.clickatell.com";
        $hw = $_GET['hid'];
        $q2 = "select news.*,students.phone,courses.`name` from
courses,students,news,courses_has_sections,courses_has_teachers where news.id='$hw' and courses_has_sections.id
= news.teacher_id and courses_has_teachers.courses_has_sections_courses_id = courses_has_sections.courses_id
and courses_has_teachers.courses_has_sections_sections_id = courses_has_sections.sections_id and students.id =
courses_has_teachers.students_id and courses.id = `courses_has_sections`.`courses_id`";
```

```

$text = "";
$to = "";
$r2 = mysql_query($q2);
if (!$r2)
{
    echo(mysql_error());
}
while($row = mysql_fetch_array($r2))
{
    if ($text == "")
    {
        $exp_date=$row['expire_date'];
        $expire_date=strftime("%d/%m/%Y", $exp_date);
        $text = "New HW: ".$row['title']." (".$row['name'].") , Due_Date: ".$expire_date;
    }
    if($to == "")
    {
        $to = $row['phone'];
    }
    else
    {
        $to = $to.", ".$row['phone'];
    }
}

// auth call
$url =
"$baseurl/http/auth?user=$clickatell_user&password=$clickatell_password&api_id=$clickatell_api_id";
// do auth call
$ret = file($url);
// split our response. return string is on first line of the data returned
$sess = split(":", $ret[0]);
if ($sess[0] == "OK")
{
    $sess_id = trim($sess[1]); // remove any whitespace
    $url
    = "$baseurl/http/sendmsg?user=$user&password=$pass&api_id=3042295&session_id=$sess_id&to=$to&text=".urlencode
    ($text)."&from=SBox";

    // do sendmsg call
    $ret = file($url);
    $send = split(":", $ret[0]);
    if ($send[0] == "ID")
    echo ("<br><br><br><img src='done.jpg' width='43' height='33' /><br><br><br>SMS
Notification sent Succesfully.");
    else
    echo "send message failed<br>". $url.'<br>'. $send[1]. $clickatell_api_id;
} else
{
    echo ("<img src='error.jpg' width='50' height='50' /><br>Authentication failure<br><br><br>
Unable to sent SMS to the Sepecified Phone numbers ,Please contact the System
Administator.");
    exit();
}
}
else
{
    echo ("<img src='error.jpg' width='50' height='50' /><br>Authentication failure<br><br><br>
This Gateway is not supported now!!!");
    exit();
}
}
?>

```

**For more information about the code lease refer to Clickatell manuals .**

## Introduction to WML, Apache, and PHP

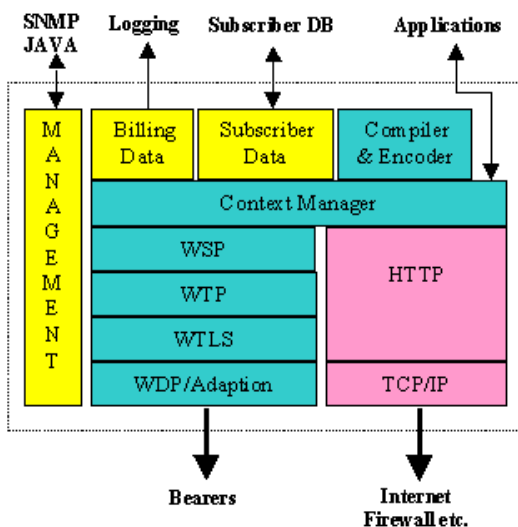
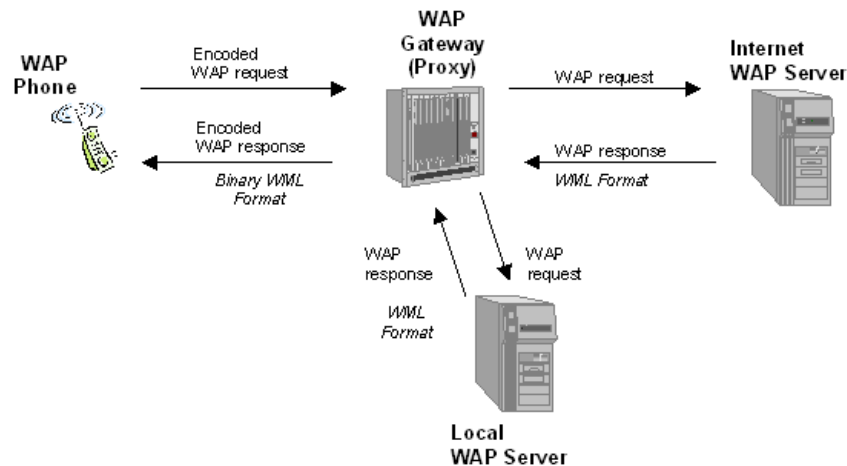
### Requirements

You will need a little experience with the Apache Web Server, Php and html. As for system requirements, I am using apache 1.3.9, php3, and Windows for this demonstration. I don't see any problems with setting this up on a Linux machine running Apache, and PHP3 or PHP4.

### Intro to WML

WML stands for Wireless Markup Language. WAP phone or similar devices are used to view pages written in WML. WML is similar to XML based on it's syntax and strictness. Anybody who has used html will have no problem learning WML. Many tags and attributes are the same, but there are fewer tags. WML allows the programmer to use variables that allow you to create dynamic content, although for our project we used php as the dynamic language ,the fig below shows the regular wap operation .

Diagram Showing WAP user request & response



WAP provides multiple applications, for business and customer markets such as banking, corporate database access, and a messaging interface

The request from the mobile device is sent as a URL through the operator's network to the WAP gateway, which is the interface between the operator's network and the Internet (see Figure 3).

Figure 3. Architecture of the WAP Gateway

## Architecture of the WAP Gateway

### WDP

The WAP datagram protocol (WDP) is the transport layer that sends and receives messages via any available bearer network, including SMS, USSD, CSD, CDPD, IS-136 packet data, and GPRS.

### WTLS

Wireless transport layer security (WTLS), an optional security layer, has encryption facilities that provide the secure transport service required by many applications, such as e-commerce.

### WTP

The WAP transaction protocol (WTP) layer provides transaction support, adding reliability to the datagram service provided by WDP.

### WSP

The WAP session protocol (WSP) layer provides a lightweight session layer to allow efficient exchange of data between applications.

### HTTP Interface

The HTTP interface serves to retrieve WAP content from the Internet requested by the mobile device.

WAP content (WML and WMLScript) is converted into a compact binary form for transmission over the air (see *Figure 4*).

The WAP microbrowser software within the mobile device interprets the byte code and displays the interactive WAP content

## WML Basics

In WML you can use many sub-pages (called 'cards') in one WML page (called a 'deck'). Each WML card works like a web page and its content is displayed to the user. The following will be our first example of a .wml page. On my server, I saved this file in ~/wireless/home.wml.

```
<wml>
<card id='home'>
<p>
My first Test page
</p>
</card>
</wml>
```

Unlike HTML, if you do not close your tags, e.g. <wml> </wml> then, your script will not compile correctly. The example above will create a simple test message on any wireless device that says "My first Test page".

## Setting up Apache

Ok, now the fun part. In order for apache to catch what a wireless device visits your server, you need to set up your httpd.conf ( my file is located in /etc/httpd/conf/ ) file, I am using PHP3 so all changes are made accordingly. Before you start you may want to make a backup copy of your httpd.conf file, just in case :)

**Step 1:** First, we need to use the AddType function to add a new mime type. You need to go to the script where you define php. It looks something like below:

```
<IfModule mod_php3.c>
  AddType application/x-httpd-php3 .php3 .php .phtml
  AddType application/x-httpd-php3-source .phps
</IfModule>
```

Needs to be changed to:

```
<IfModule mod_php3.c>
  AddType application/x-httpd-php3 .php3 .php .phtml .wml
  AddType application/x-httpd-php3-source .phps
</IfModule>
```

This will use the php compiler on all .wml pages it encounters.

**Step 2:** Uncomment the apache Load Module and Add Module change:

```
#LoadModule rewrite_module      modules/mod_rewrite.so
#AddModule mod_rewrite.c
```

to:

```
LoadModule rewrite_module      modules/mod_rewrite.so
AddModule mod_rewrite.c
```

**Step 3:** We'll use apache mod\_rewrite module (only available in version 1.2+ ). Using this, you can rewrite requested URL's on the fly, as certain conditions are met. You need to place this code snippet at the bottom of the page.

```
RewriteEngine On
# Catch most WAP browsers
RewriteCond %{HTTP_ACCEPT} text/vnd\.wap\.wml [OR]
# WinWAP, WAPjag
RewriteCond %{HTTP_USER_AGENT} wap [OR]
#Nokia emulators (sdk)
RewriteCond %{HTTP_USER_AGENT} 7110
# Rewrite to where your wireless page is located
RewriteRule ^[\./](.*)$ /home/mydirectory/wireless/home.wml [L]
```

Now, you will need to restart the apache server.