NORWEGIAN UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY, MATHEMATICS AND ELECTRICAL ENGINEERING

MASTER'S THESIS

Student's name:	Bent Erik Skogstad
Area:	Telematics
Title (Norwegian):	**MRS – Mobile Repository System: Nettsentrisk synkronisering og lagring av data**
Title (English):	**MRS – Mobile Repository System: Net-centric data synchronization and storage**

Description: There are numerous mobile devices that handle address data, notes, calendar information etc. Synchronization of such data between different devices is on the other hand quite cumbersome. A mobile phone and a PDA synchronize their data with a computer, which assumes a location dependent operation. The assignment is to design and implement a net-centric system for storing information to be downloaded to a mobile phone and a PDA. The devices must be able to update the system, also through a web based interface.

The system is to be implemented using standard protocols from OMA and an open source implementation of a server, for instance Sync4j. MRS shall communicate with mobile terminals using SyncML DS over GPRS or WLAN. From mobile terminals the user shall be able to request/search for information in MRS (for instance a company's employer address list, documents etc.). Client, user interface and server for this system are to be implemented.

MRS will be realized in the PATS lab and piloted for the employers at Telenor R&D. If possible, it is wanted that a pilot will be run for the employers at St. Olav's Hospital. Challenges facing this pilot will be that the employers will be using IP telephones and MDAs in their daily practice, but outside the hospital area they will use mobile phones. MRS must be able to handle the fact that a user has several different types of mobile terminals, but that all of them should have the same contact information.

The assignment shall also discuss architecture, scalability, reliability, performance and openness. Business models shall be developed, including actor/role-analyses dealing with such a service realized in an internal network, as an internal service delivered by Telenor or a third party or as a regular public service delivered by Telenor or a third party. SWOT-analysis and system-interface considerations, especially regarding transport through closed networks, shall be worked out.

Start date:	19 January 2005
Deadline:	15 June 2005
Submission date:	13 June 2005
Department:	Department of Telematics
Supervisors:	Anne Marte Hjemås and Frode Flægstad, Telenor R&D

Trondheim, 22 February 2005

Lill Kristiansen,
Professor

Preface

This thesis concludes my Master of Science education in Communication Technology at The Norwegian University of Science and Technology (NTNU) in Trondheim. The thesis was performed throughout my 10th semester, spring 2005, at the Department of Telematics, in collaboration with Telenor R&D.

I would like to thank professor Lill Kristiansen and supervisors Anne Marte Hjemås and Frode Flægstad, both from Telenor R&D, for valuable ideas and comments throughout the work period. A thank also goes to the users of the Sync4j-users mailing list for their help troubleshooting various technical problems.

Trondheim, 13 June 2005

Bent Erik Skogstad

Table of contents

List of figures

List of tables

Abbreviations

CGI	Common Gateway Interface
DS	Data Synchronization
DTD	Document Type Definition
GPRS	General Packet Radio Service
GUID	Globally Unique Identifier
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IMEI	International Mobile Equipment Identity
ISP	Internet Service Provider
JSP	JavaServer Pages
LUID	Locally Unique Identifier
MRS	Mobile Repository System
OMA	Open Mobile Alliance
PDA	Personal Digital Assistant
PIM	Personal Information Manager
SLA	Service Level Agreement
SWOT	Strengths, Weaknesses, Opportunities and Threats
SyncML	Synchronization Markup Language
WebDAV	Web-based Distributed Authoring and Versioning
WLAN	Wireless Local Area Network
WML	Wireless Markup Language
URI	Uniform Resource Identifier
XML	Extensible Markup Language

Abstract

This thesis presents a net-centric solution that offers users the possibility to synchronize their contact and event data with a server and access data from a central data repository, independent of the location of the user. The data in this repository can be retrieved and modified through user interfaces for mobile phones, PDAs and computers.

The devices synchronize their data using the SyncML synchronization protocol, an open standard for data synchronization. The server used is the Sync4j SyncServer, and it is a Java based SyncML server that can be used with any SyncML enabled client. Since synchronization is a central aspect, the report will present basic principles for data synchronization. It will also provide a technical overview of SyncML and the Sync4j implementation.

The main implementation focus is directed towards user functionality for adding, modifying, deleting and searching for contacts and events in the central repository. The report will describe these features for the different device types.

Further, the report will identify roles that need to be covered when deploying the service in a market, and it will also look at business relationships between these roles. Three scenarios for deploying the service are investigated, and from these one can observe how the different roles change, depending on how the service is realized. The scenarios will look at the service realized in an internal network, as an internal service offered by a retailer or a third party, or as a public service, and these scenarios are illustrated by small cases to point out the relations between actors and roles.

Finally, aspects concerning architecture, scalability, reliability, performance, openness and system/interface are looked into.

1 Introduction

The introduction chapter will give a brief description of the background and motivation for this master's thesis, pointing out why there is a need for net-centric data synchronization and storage. Scope and limitations for the thesis will be defined, and finally, the chapter outline of this report is presented.

1.1 Background and motivation

With a growing amount of mobile devices for both private and professional use, there is a greater need to manage contact data, calendar information, notes, tasks etc. This information needs to be consistent, in the sense that a user should have access to the same, correct data independently of which device he/she is currently using. Synchronization provides a means to achieve this consistency. Traditionally, synchronization has taken place between for instance a computer and the different mobile devices directly, but this operation is location dependent since the user has to be physically near the computer to be able to synchronize. A location independent solution would therefore be of much greater value, and this is one of the main drives behind this thesis. The idea is that mobile devices can access a centralized server that stores the necessary information, and in that way be able to synchronize their content remotely.

It is not always practical that the user synchronizes his/her device with the entire content of the server. Nevertheless, the user still should have the possibility to get important information from the server and also update the central data. Therefore, it is desirable that there exists user interfaces and functionality for retrieving this information. In addition to data synchronization, this is also an important drive behind the thesis.

1.2 Scope and limitations

The data type considered in this report is related to contacts and calendar events. The thesis focuses mainly on developing functionality for mobile phones, PDAs and computers. More specifically, the functionality is related to the addition, modification and deletion of contact and event items, as well as searching for data among these items. User interfaces are necessarily required to demonstrate the functionality, but developing these interfaces is not the main area of the thesis.

The assignment description implies that SWOT analyses are to be worked out. This has not been the focus of the work in this report. Instead, the author has directed more efforts towards other business aspects, such as business relationships and actor/role considerations.

1.3 Thesis outline

Chapter 2 presents the most important principles for data synchronization. These include ID handling, change detection, conflict detection and resolution, slow and fast synchronization and synchronization topologies and types.

Chapter 3 introduces the SyncML framework, describes the Sync4j implementation of a SyncML server and explains the vCard and iCalendar format associated with contact and event details, respectively.

Chapter 4 presents some scenarios and use cases. These will be used to gather requirements for the system.

Chapter 5 describes the implemented system. This involves a description of the server, user interfaces and functionality for both web and WAP, as well as system tests.

Chapter 6 contains business considerations for various service realization scenarios, focusing on describing business roles.

Chapter 7 discusses important topics related to the system implementation. These topics are architecture, scalability, reliability, performance, openness and system/interface considerations. This chapter also points out areas for future work.

Chapter 8 presents conclusions.

2 Data synchronization principles

There are a number of issues relating to data synchronization that need to be considered. The most important of these issues will be presented in this chapter, and these include ID handling, change detection, conflict detection and resolution, slow and fast synchronization and synchronization topologies and types.

2.1 ID handling

The data records need to be uniquely identified by an identifier. The identifiers of the records residing on the central server are called GUIDs (globally unique identifiers) and the record identifiers on the various devices are called LUIDs (locally unique identifiers) [1]. These identifiers point to the same items, but are not necessarily the same on all devices and server, and for that reason the server needs to have a mapping table between the GUIDs on the server and the LUIDs on the client devices, see Table 2.1.

GUID	Device A LUIDs
1	101
2	102
3	103

GUID	Device B LUIDs
1	201
2	202
3	203

Table 2.1: LUID-GUID mapping (based on [2])

The GUID entries are the server's identifiers, and a mapping between these GUIDs and each device's LUIDs is necessary.

It is also possible that the server and devices generate GUIDs that must be used by all other devices. This eliminates the need of a mapping table, since all of the client LUIDs for an item will be equal. An ID scheme that defines a convention on how to generate identifiers may also be agreed upon by the clients and the server. Other approaches may also be considered.

2.2 Change detection

Change detection involves identifying what data that have changed since the last synchronization [1]. Additional information associated with each record, such as timestamps and state information, can be used for this purpose. Timestamps indicate the point in time of the last synchronization, and the state information could indicate if the record is new, updated or deleted. Table 2.2 illustrates a possible table in a database used for change detection.

id	state	last_update
1	N	2005-05-01 18:24
2	U	2005-05-02 08:30
3	D	2005-05-02 08:35

Table 2.2: Example database change table (modified from [1])

The table contains the last update of each item, and the state indicates whether the item is new (N), updated (U) or deleted (D).

If these types of additional information are not available, more complicated approaches must be used, such as comparing the contents of an item, field by field.

2.3 Conflicts

One of the main challenges with data synchronization is the arising of conflicts. Due to the fact that the same data resides on multiple devices, it is likely that the same entry may be changed on more than one device. Aspects related to conflicts are presented here according to [2].

2.3.1 Conflicting updates

A write-write conflict occurs when the same field is updated concurrently in two different copies of the datastore.

A read-write conflict is due to a stale read, a situation where one receives an entry from one copy of the datastore that is not up to date.

A third type of conflict is constraint violation. To illustrate, a constraint that only allows numbers between 0 and 100 is changed, so that now only numbers between 0 and 50 are allowed. In another copy of the datastore, a user enters a value between 50 and 100, which still is in accordance with the old constraint rule, but in clear violation to the updated rule.

2.3.2 Conflict detection

All records are uniquely identified by an identifier, and the central server keeps a list of mappings between the identifiers used by the server and the identifiers used by the client(s). Synchronization usually involves the client sending a list of the records that have changed since the last synchronization. The server also generates a list of records that have changed in the same time period, and then compares the two lists. Every record that exists in both lists is identified as a conflict and resolved according to certain preferences, described next.

[1] presents a synchronization matrix for two databases, database A and database B, see Table 2.3. This table illustrates the actions to be performed whenever an item changes in one or both of the databases.

Table 2.3: Synchronization matrix (from [1])
The table illustrates in which situations conflicts arise, as well as actions to be performed in case of changes that does not lead to conflicts. Grey cells means that the items are either equal or that no action is necessary.

Database A → ↓ Database B	**New**	**Deleted**	**Updated**	**Synchronized/ Unchanged**	**Not Existing**
New	Conflict	Conflict	Conflict	Conflict	B replaces A
Deleted	Conflict		Conflict	Delete target	
Updated	Conflict	Conflict	Conflict	B replaces A	B replaces A
Synchronized/ Unchanged	Conflict	Delete target	A replaces B		B replaces A
Not Existing	A replaces B		A replaces B	A replaces B	

2.3.3 Conflict resolution

As can be seen from Table 2.3, several change combinations can lead to conflicts. There are many ways to resolve these conflicts. *Manual conflict resolution* means that entries are simply duplicated and marked as conflicts, leaving the resolution to the user. *Automatic conflict resolution* is on the other hand done by the server based on user preferences, meaning that one of the conflicting records is chosen as winner and the other one is deleted. The preferences can for instance be:

- Client side updates always win
- Server side updates always win

- Latest change wins

A third type of conflict resolution is merging the conflicting records into a new one. For this to work, the server needs to analyze the structure of the record and be able to identify which fields that have changed.

2.4 Slow versus fast synchronization

Fast synchronization involves synchronizing only the items that have changed since the last synchronization [1]. The precondition of this synchronization mode is that the last synchronization was completed successfully. If this condition is not fulfilled, for example due to a device reset, slow synchronization is necessary. In this mode, the client has to send its entire database to the server. The server then compares it with its own database and sends the modifications back to the client. When two devices are synchronizing for the first time, slow synchronization must be used.

2.5 Usage modes

This section presents the three physical modes in which synchronization is typically performed, according to [2]. These modes are local, pass-through and remote.

2.5.1 Local

The local mode typically involves a computer and a mobile device that is connected to this computer via a serial, infrared, USB or Bluetooth connection. The server is an application running on the computer, and the data can be retrieved from applications such as Lotus Notes and Microsoft Outlook.

2.5.2 Pass-through

This mode is similar to the local mode since it also means that a mobile device is connected to a computer. But instead of the computer functioning as the synchronization server, the device's request is passed forward to the real synchronization server. It is often beneficial to have the data to be synchronized stored at a remote location instead of locally, and this mode facilitates such a scheme. On the other hand, the user must synchronize with a remote server even though he/she is not on the road, generating some additional network traffic compared to the local synchronization mode.

2.5.3 Remote

In the remote synchronization mode, there is no pass-through synchronization server in the middle. The mobile device establishes a network connection directly with the real synchronization server. Remote synchronization and pass-through synchronization can successfully be used in combination. At the office, the user can benefit from a cheaper and faster network connection using the pass-through mode, and out of office the user can still connect to the synchronization server. The remote synchronization mode will be the focus of this report.

2.6 Topologies

There are various synchronization topologies that logically define different ways of synchronizing data. The main topologies are [2]:

- One-to-one
- Many-to-one
- Many-to-many
- Cluster and hierarchy

These topologies are illustrated in Figure 2.1.

2.6.1 One-to-one

In this topology there are only one server and one client, and for this reason this scheme is also called dedicated pair. If the data is changed only on the client side, there will be no conflicts using this topology. When data is changed on both the server and client sides, possible conflicts can for instance be identified by the server and resolved.

2.6.2 Many-to-one

The many-to-one topology involves a central server, and the data is exchanged directly between this server and a set of clients. Two clients cannot exchange data directly, and conflicts can therefore only occur at the central server, which then resolves them accordingly.

2.6.3 Many-to-many

In the many-to-one topology, the central server might be a system bottleneck. In the many-to-many topology there is no central server, in the sense that every client also acts as a server. All clients must be able to get updates and send updates to every other client. This is the most complex topology, since all the clients must be able to detect and resolve conflicts. On the other hand, this topology is the most robust, since you eliminate the bottleneck that exists in the many-to-one topology.

2.6.4 Cluster and hierarchy

The cluster and hierarchy topologies can be seen as hybrids of the many-to-one and many-to-many topologies, as described in the two previous chapters. In a *cluster* topology, there are a cluster of servers which all contain copies of the data and replicate between each other. Only one of the servers in the cluster keeps the authoritative copy. The servers may be geographically distributed to reduce the distance between server and clients. In a *hierarchy* topology, the servers are organized in a hierarchical manner, with the servers at one level at the same time acting as clients of a server on the level above. In both topologies, the overall system will not be affected by a failure in one of the servers.

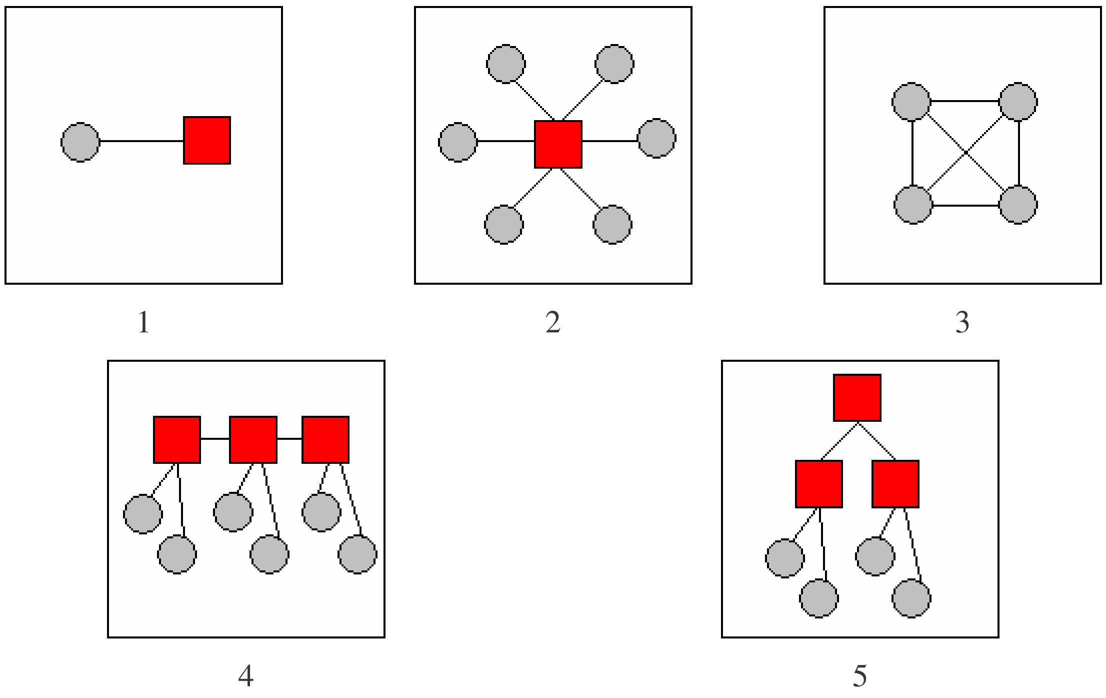


Figure 2.1: Synchronization topologies (from [2])

1: One-to-one. 2: Many-to-one. 3: Many-to-many. 4: Cluster. 5: Hierarchy. (Circles represent clients and squares represent servers).

3 Technology overview

In this chapter, central technologies, formats and implementations will be described in further detail. The SyncML specification, which is an open specification for data synchronization, is presented first. The open source initiative Sync4j, implementing the SyncML protocol, is a central part of the work done in this assignment, and will be described in the following chapter. Finally, the standard formats for representing contacts and calendar events, vCard and iCalendar, are introduced.

3.1 SyncML

The SyncML Initiative was founded in February 2000 by the founding sponsors Ericsson, IBM, Lotus, Motorola, Palm, Psion and Starfish [2]. The first version of the specification was released December 2000. The initiative has now consolidated into the Open Mobile Alliance (OMA), and it is separated in two working groups: Device Management Working Group and Data Synchronization Working Group [3]. This assignment will concentrate on data synchronization, and will not focus on device management.

The two fundamental parts of this specification are the SyncML Synchronization Protocol [4] and the SyncML Representation Protocol [5]. The Synchronization Protocol describes the sequence of packages that applications must exchange in order to communicate changes to each other. The Representation Protocol describes the syntax for specifying changes an application has made to its data. Figure 3.1 shows the main elements of the SyncML Framework.

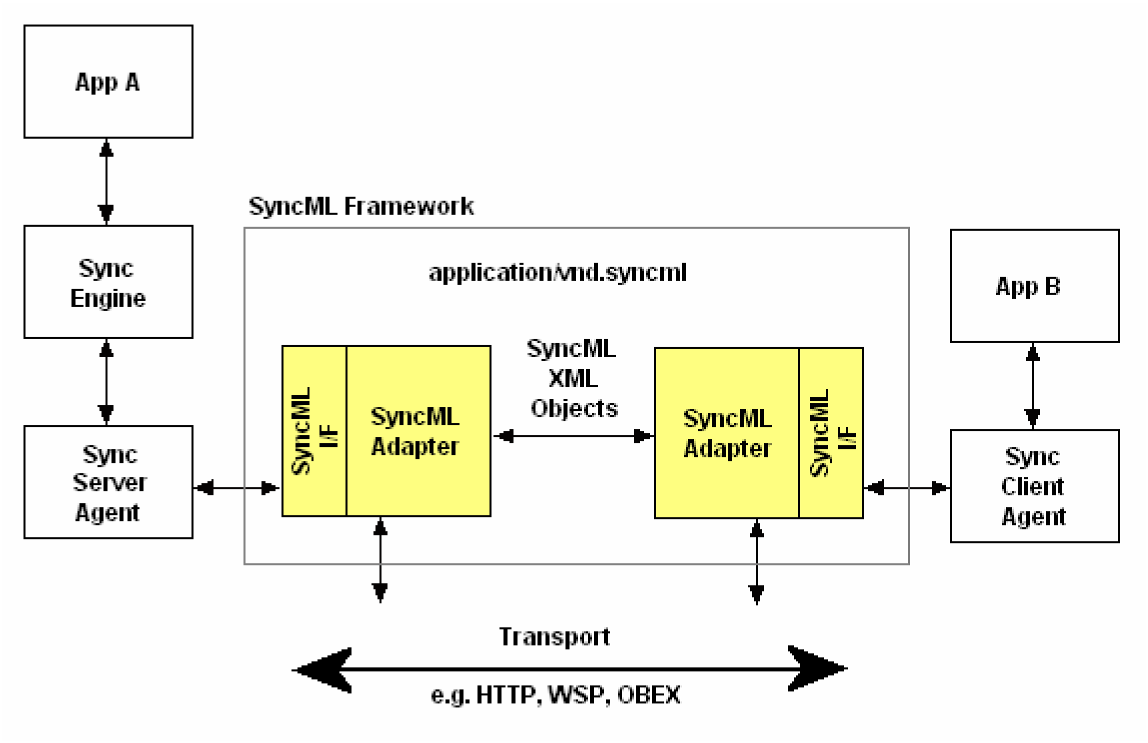


Figure 3.1: SyncML Framework (redrawn from [5])

The main elements of the SyncML framework and the supporting entities (applications, synchronization engines and synchronization agent)

App A is the server application that provides the synchronization service for client applications (App B). The synchronization engine is a logical entity that keeps track of changes made by the server application, and it also has methods for detecting and resolving changes retrieved from the clients [2]. The synchronization agent is responsible for the generation and processing of SyncML packages. The Sync Server Agent manages the Sync Engine's network access and communicates the synchronization operations to/from the client application through the SyncML I/F [5]. This is the interface towards the SyncML Adapter, which the originator and recipient of SyncML formatted objects use to communicate with each other. The SyncML Adapter also interfaces with the transport layer. On the client side, the Sync Client Agent accesses the network, as well as the SyncML Adapter (through the SyncML I/F).

3.1.1 SyncML messages

During synchronization, packages are logically exchanged between the entities [2]. A logical package can be partitioned into multiple physical messages. A SyncML message is specified using XML. The message header includes source, target, routing, session and authentication information. The body is a set of SyncML commands identifying synchronization operations such as add, delete, replace, search, status etc. Each command refers to specific data items. The SyncML commands are listed in [5].

Figure 3.2 shows an initialization SyncML message. Lines 2 to 14 form the header, and lines 15 to 26 constitute the body. The header contains DTD and protocol version information, hhsession ID, message ID, target and source information. A `<Cred>` element (lines 9 to 12) indicates user credentials, in this case using basic authentication. The `<MaxMsgSize>` element indicates the maximum SyncML message size that the client can receive.

The `<Alert>` lines (16 to 24) tell the server to synchronize with the database referred to in the server URI *./exchange-contacts* with the database *EriPBDB* on the client device. The `<Anchor>` (line 22) is used to check whether the last synchronization terminated correctly [4]. After a successful synchronization, the `<Next>` anchor is stored, and on the next session, this value is sent as `<Last>`. When the device in question stores `the` `<Next>` anchor, it is able to compare whether the sync anchor is the same as the `<Last>` anchor sent by another device. If `<Next>` and `<Last>` do not match, the last session did not terminate correctly, and slow synchronization is initiated. Otherwise, fast synchronization can be used. It is vital that the stored synchronization anchors are not updated before the synchronization session is finished.

The example in Figure 3.2 is an initialization message from the client to the server, since the `<Source><LocURI>` is a mobile phone IMEI number (line 8), and the `<Target><LocURI>` is the IP address of the server (line 7). This message and the following examples are copied from the server log file of the implementation.

```
1 <SyncML>
2   <SyncHdr>
3     <VerDTD>1.1</VerDTD>
4     <VerProto>SyncML/1.1</VerProto>
5     <SessionID>7</SessionID>
6     <MsgID>1</MsgID>
7     <Target><LocURI>http://129.241.219.132/sync4j/sync</LocURI></Target>
8     <Source><LocURI>354007004732258</LocURI></Source>
9     <Cred>
10      <Meta><Format>b64</Format><Type>syncml:auth-basic</Type></Meta>
11      <Data>ZW1wbG95ZWU6ZW1wbG95ZWU=</Data>
12    </Cred>
13    <Meta><MaxMsgSize>2700</MaxMsgSize></Meta>
```

```
  </SyncHdr>
  <SyncBody>
    <Alert>
      <CmdID>1</CmdID>
      <Data>200</Data>
      <Item>
        <Target><LocURI>./exchange-contacts</LocURI></Target>
        <Source><LocURI>EriPBDB</LocURI></Source>
        <Meta><Anchor><Last>2152</Last><Next>2154</Next></Anchor></Meta>
      </Item>
    </Alert>
    <Final></Final>
  </SyncBody>
</SyncML>
```

Figure 3.2: Initialization SyncML message (from log file)

The next figure, Figure 3.3, shows how an actual synchronization command is represented. The root of the synchronization is the `<Sync>` tag. This tag has `<SyncBody>` as its parent element, `and` contains the command ID, target and source information, and which synchronization operations that will be executed. The most common are add, replace and delete. In this example, the `<Add>` command (lines 5 to 19) specifies that a new item is to be added, and a vCard representation of an item is given in the `<Data>` tag (lines 10 to 17).

```
 1 <Sync>
 2   <CmdID>3</CmdID>
 3   <Target><LocURI>EriPBDB</LocURI></Target>
 4   <Source><LocURI>./exchange-contacts</LocURI></Source>
 5   <Add>
 6     <CmdID>2</CmdID>
 7     <Meta><Type>text/x-vcard</Type></Meta>
 8     <Item>
 9       <Source><LocURI>cbb0135ac6c3894b8816f93031c606c20000000130ac</LocURI></Source>
10       <Data>
11         BEGIN:VCARD
12         VERSION:2.1
13         N:Skogstad;Bent Erik;;;
14         FN:Skogstad, Bent Erik
15         TEL;CELL:+4712345678
16         END:VCARD
17       </Data>
18     </Item>
19   </Add>
20 </Sync>
```

Figure 3.3: Add command (from log file)

Let us say that this item is being modified. Instead of the `<Add>` tag there would be a `<Replace>` tag in the `<Sync>` section, see Figure 3.4 (lines 5-23). The example shows that the vCard has been modified on the server, since the target database is on the client (line 3) and the source database is on the server (line 4).

```
 1 <Sync>
 2   <CmdID>3</CmdID>
 3   <Target><LocURI>EriPBDB</LocURI></Target>
 4   <Source><LocURI>./exchange-contacts</LocURI></Source>
 5   <Replace>
 6     <CmdID>2</CmdID>
 7     <Meta><Type>text/x-vcard</Type></Meta>
 8     <Item>
 9       <Target><LocURI>0000000003A3</LocURI></Target>
10       <Data>
11         BEGIN:VCARD
12         VERSION:2.1
13         N:Skogstad;Bent Erik;;Student;
14         FN:Skogstad, Bent Erik
15         ADR;HOME:;;Weidemanns vei 17;Trondheim;;7014;Norway
17         TEL;CELL:+4799608303
18         EMAIL;INTERNET:benterik@stud.ntnu.no
20         END:VCARD
21       </Data>
22     </Item>
23   </Replace>
24 </Sync>
```

Figure 3.4: Replace command (from log file)

If the item is deleted on the server side, the tag `<Delete>` in Figure 3.5 will be included in the `<Sync>` section (lines 5 to 8). The `<Target><LocURI>` tag is a reference to the item to be deleted on the mobile phone (line 7).

```
 1 <Sync>
 2   <CmdID>3</CmdID>
 3   <Target><LocURI>EriPBDB</LocURI></Target>
 4   <Source><LocURI>./exchange-contacts</LocURI></Source>
 5   <Delete>
 6     <CmdID>2</CmdID>
 7    <Item><Target><LocURI>0000000003A3</LocURI></Target></Item>
 8   </Delete>
 9 </Sync>
```

Figure 3.5: Delete command (from log file)

3.1.2 Synchronization types

[4] specifies seven different synchronization types:

Two-way synchronization
The client and the server exchange information about modified data, with the client sending its modifications first.

Slow synchronizatoin
With slow sync not just the modified data are compared, but all items are compared on a field-by-field basis.

Refresh synchronization from client only
The client sends all of its data to the server, and the server is expected to replace all data in the target database with the data sent by the client, i.e. the client has the right of way.

Refresh synchronization from server only
Similar to the above synchronization type, except that it is the *server* that sends all of its data to the client, and the client is expected to replace all data, i.e. the server has the right of way.

One-way synchronization from server only
The client gets all modifications from the server, but the client does not send any of its modifications to the server.

Server alerted synchronization
The server alerts the client to perform synchronization.

3.2 The Sync4j Project

The Sync4j Project is an open source initiative to deliver a complete mobile application platform implementing the SyncML protocol [6]. The server, SyncServer, is a Java based SyncML server that can be used with any SyncML enabled client.

3.2.1 Sync4j SyncServer

This chapter describes the Sync4j SyncServer. All of the information given is taken from [1].

The SyncServer architecture is layered, as illustrated by Figure 3.6. The *transport layer* is where the client messages reach the system. Currently, the HTTP protocol and binding are implemented. Other transport protocols may be added. The *protocol layer* handles the interpretation and handling of the SyncML protocol. Other synchronization protocols may be added. The *server layer* is the implementation of the synchronization server. The *application layer* implements the server's interaction with external environments. The *framework* implements and provides services and abstractions used by different layers to implement the component they are built of. The most important of these services are Core SyncML representation and protocol, configuration framework, logging framework, SyncML DS engine framework, security framework and other commonly used utilities. For additional details, the reader is referred to [1].

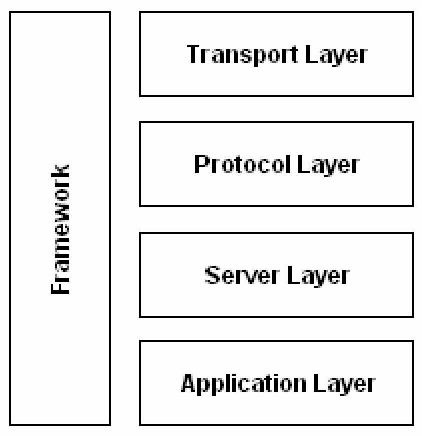


Figure 3.6: Sync4j SyncServer layered architecture (from [1])

In addition to being a layered architecture, the SyncServer has a modular design. Table 3.1 presents the main modules that build up the Sync4j SyncServer.

Table 3.1: Sync4j SyncServer modules

Module	Description
Sync4j Engine	The core of the SyncServer. Contains synchronization logic needed for: • identifying sources and destinations of datasets to be synchronized • identifying what data needs to be updated, added or deleted • determining how the updates must be applied • conflict detection and resolution
Transport Layer module	Implements the transport specific binding of SyncML.
SyncML module	Responsible for encoding and decoding SyncML messages
Protocol	Implements the SyncML synchronization protocol
Services module	Provides horizontal services, such as authentication, security, configuration, logging etc.
SyncSources	Facilitates integration with external and legacy systems

3.2.2 Execution flow

When a device sends a request to the server, the execution flow is as shown in Figure 3.7.

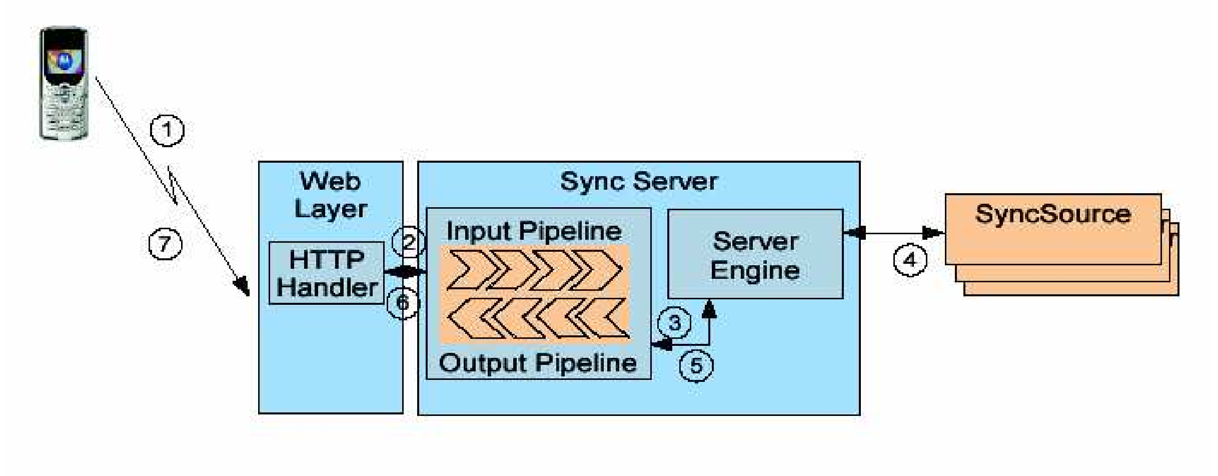


Figure 3.7: Execution flow (from [1])

The figure shows the execution flow of a client request. This request is processed by the SyncServer and a response is sent back to the client.

1. The device sends the request and the HTTP handler processes it, e.g. associating it with an ongoing session. The processed request is sent to the SyncServer.
2. The input pipeline of the SyncServer is a preprocessing instance that can be customized to process the messages according to specific application needs.

3. The processed message is sent to the server engine, which does the actual synchronization processing.
4. The engine may use custom sync sources to access external data stores, if necessary.
5. The engine then builds a response that goes through the output pipeline for postprocessing.
6. The HTTP handler takes the messaged processed in the output pipeline and builds a HTTP response.
7. Finally, the HTTP message is sent back to the device.

3.3 The vCard format

vCard is an electronic representation of a person's business card. A number of information types can be specified. The latest vCard version is version 3.0, specified in [7]. The version used in the Sync4j implementation is the previous version, version 2.1, and therefore this section will refer to this version. vCard version 2.1 is specified in [8].

A vCard 2.1 representation begins with `BEGIN:VCARD` and `VERSION:2.1`, and ends with `END:VCARD`. Table 3.2 only lists a subset of the information types defined in [7], but the most important types, in the author's opinion, are covered.

Table 3.2: vCard information types

Information type	Format
Name	`N:`*`Surname`*`;`*`First name`*`;`*`Middle name`*`;`*`Salutation`*`;`*`Suffix`*
Formatted name	`FN:`*`Formatted name`*
Organization	`ORG:`*`Company`*`;`*`Department`*
Job title	`TITLE:`*`Job title`*

Telephone numbers	`TEL;WORK;VOICE:`*`business phone number`*
	`TEL;HOME;VOICE:`*`home phone number`*
	`TEL;CELL;VOICE:`*`mobile phone number`*
	`TEL;HOME;FAX:`*`home fax number`*

Work address	`ADR;WORK:;;`*`Street`*`;`*`City`*`;`*`County`*`;`*`Postal code`*`;`*`Country`*
Home address	`ADR;HOME:;;`*`Street`*`;`*`City`*`;`*`County`*`;`*`Postal code`*`;`*`Country`*
Role	`ROLE:`*`Role`*
Birthday	`BDAY:`*`Birthday`*
E-mail	`EMAIL;PREF;INTERNET:`*`e-mail address`*
Revised	`REV:`*`Date of revision`*

An example vCard is shown in Figure 3.8.

```
BEGIN:VCARD
VERSION:2.1
N:Nordmann;Kari;A.;Ms;d.y.
FN:Nordmann, Kari A.
NICKNAME:Karri
ADR;HOME:;;Havnegata 1;Oslo;Oslo;0123;Norway
BDAY:1950-01-01
TEL;CELL:+4734567890
TEL;VOICE;HOME:+4712345678
TEL;FAX;HOME:+4723456789
EMAIL;INTERNET:kari.nordmann@sykehuset.no
ADR;WORK:;;Sykehusgata 1;Oslo;Oslo;0123;Norway
ROLE:Coordinator
TITLE:Secretary
ORG:The Hospital;Administration
TEL;VOICE;WORK:+4745678901
NOTE:Eksempelperson
REV:20041020T142542
END:VCARD
```

Figure 3.8: Example vCard

3.4 The iCalendar format

iCalendar is an electronic representation of events. A number of information types can be specified. An iCalendar representation begins with `BEGIN:VCALENDAR` and `BEGIN:EVENT`, and ends with `END:VEVENT` and `END:VCALENDAR`. The information types in the example iCalendar in the example iCalendar in Figure 3.9 are a subset of the information types defined in [9], but the most important types, in the author's opinion, are covered. The fields are reasonably self-explanatory and will not be described in further detail.

```
BEGIN:VCALENDAR
BEGIN:VEVENT
CATEGORIES:Internal
DESCRIPTION:Staff meeting
LOCATION:London
SUMMARY:Meeting
DTEND:20050805T080000Z
DTSTART:20050805T060000Z
UID:1115730846226[1]
END:VEVENT
END:VCALENDAR
```

Figure 3.9: Example iCalendar

[1] UID (Unique Identifier): This property defines the persistent, globally unique identifier for the event [9]

4 Requirements

This chapter presents some scenarios that illustrate synchronization needs and also the use of storing central company data. Not all desired features described in the scenarios are implemented, but the scenarios are still included to present ideas for future work that could be useful. The main purpose of describing scenarios is to gather system requirements, and these will be presented for each scenario, and summarized formally at the end of the chapter.

4.1 Scenarios

Scenarios are first given to illustrate, followed by a textual use case description that describes the flow of actions needed to realize the scenario. These scenarios will yield a number of requirements to the system.

4.1.1 Loss of mobile phone

Mary was on holidays in southern Norway and was unfortunate enough to lose her mobile phone in the water. For a while she had planned to buy a new phone, so she was not very upset. After she bought the new phone she realized that she also had lost her entire contact list. Luckily, Mary was a Telenor customer and had her list of contacts stored on a central server. The only thing she had to do was to synchronize her mobile phone with the server, and shortly after she once again had her contact list intact.

Table 4.1: Use case 1 – Transfer contact list to new phone

Use case 1	**Transfer contact list to new phone**
Use case description	The user has lost her mobile phone, and synchronizes the contact list on her new phone with a central server
Actor	Mobile phone end user
Flow	
1	The user logs on the central server with her new mobile phone
2	The user chooses to start synchronization of contacts
3	Since the new mobile phone does not have any contacts, all the user's contacts will be copied from the central server to the mobile phone

Requirements:

- The server shall function as a backup storage, so that a user can retrieve all of his/her contacts if the phone is lost or damaged

4.1.2 Contact list lookup

Henry is a doctor at the Department of Neuro Surgery at St. Olav's Hospital, and is stuck on a lung related problem. He does not know any lung specialists, but his mobile phone can help him get in touch with one. He performs a search in the central database after doctors related to the Department of Heart and Lung Surgery, who are not busy with other appointments. The search yields only one result – Catherine Eckhoff. He calls her and explains the problem, and she is able to give him an answer over the phone. After the conversation Henry thinks to himself that he might need to call her back at a later time, and saves her contact information on his own mobile phone.

Table 4.2: Use case 2 – Contact list lookup

Use case 2	**Contact list lookup**
Use case description	The user searches for a person that has a particular role/title or group/department membership.
Actors	Employee
Flow	
1	The user logs on the central server
2	The user chooses to search for a contact
3	The user specificies the title/role of the person he is looking for, and/or what group/department he/she belongs to. This information can be chosen from a list. The user indicates if the person must be available at the present time (no appointments in the calendar).
4	The server returns the search results.
5	The user chooses one of the persons to save locally.

Requirements:

- Central database must include role/title, group/department membership etc.
- It shall be possible to display only available people, i.e. persons who do not have any appointments in their calendar at the time of the lookup
- From the list of results it shall be possible to save contact information locally

4.1.3 New contact information

Michael and Sarah are new consultants in Top Consulting, and receive their own PDA and mobile phone. These devices are being synchronized so that the company's list of contacts is saved locally on their devices. Betty in the reception adds the new phone numbers in the central employee database. Afterwards the database is synchronized with the devices of all employees in Top Consulting, so that they can reach Michael and Sarah.

Table 4.3: Use case 3 – New contact information

Use case 3	**New contact information**
Use case description	A new contact is added to the central list of contacts, and all the contact lists on the users' devices are synchronized with the server.
Actor	Person in charge of personnel
Flow	
1	The user logs on the central server
2	The user chooses to add a new contact
3	The user enters all available information and group/department memberships
4	The server sends a Server Alert to the users, either to the entire system or to a relevant group/department
5	The users synchronize their devices with the server

Requirements:

- It shall be possible to add a new contact to the central database
- The system shall be able to issue server alerts when items change, requesting users to initiate synchronization

4.1.4 Calendar appointment

John is to set up a meeting with two other co-workers, Susan and Tom. He uses a laptop to log on to the synchronization server through the web interface. The system has information about the employees' calendars, and this facilitates that John is not allowed to add a new appointment that is in conflict with other Susan's and Tom's other appointments. When John has found an available time slot, the system will make sure that Susan's and Tom's mobile devices is synchronized with the server and that they are alerted about the meeting. Both Susan and Tom have to confirm the appointment.

Table 4.4: Use case 4 – Add a new appointment involving other people

Use case 4	**Add a new appointment involving other people**
Use case description	The user is to set up a meeting with co-workers, and uses the system to find an available time slot for the meeting
Actors	User A (sets up meeting), user B (participates in meeting)
Flow	
1	User A logs on to the system
2	User A chooses to add a new event in the calendar
3	User A indicates that other people will be participating, and chooses these (user B) from a list (possibly a search).
4	An overlapping view of user A's and B's calendars is displayed to user A, and user A chooses a time period that is suitable for both user A and user B. The status of the appointment will be *Pending.*
5	The server will issue a Server Alert to user B's mobile device to initiate synchronization with the server.
6	User B synchronizes his calendar with the server.
7	User B receives a message that a new appointment awaits acceptance.
8	User B accepts, and synchronizes once again with the server.
9	User A receives a message that user B has accepted, and the status of the appointment is *Active.*
Alternative flow	
8	User B is not able to attend this meeting, refuses, and synchronizes with the server.
9	User A is notified (through Server Alert) that user B has refused the suggested meeting, and user A can try to find a new time (steps 1-5).

Requirements:

- It shall be possible to choose which persons to include in an appointment
- The participants must be alerted about the appointment (through Server Alert) and they also have to confirm the suggested appointment

4.2 List of requirements

Requirements are gathered from the scenarios and more requirements are deduced from these. These, as well as requirements specified explicitly in the thesis description, are concretized and listed in Table 4.5.

Table 4.5: List of requirements

ID	Description
R1	The system shall use an open source implementation of a server
R2	The system shall use SyncML DS over GPRS and/or WLAN
R3.1a	The user shall be able to synchronize the contact list on a mobile phone with the server
R3.1b	The user shall be able to synchronize the calendar on a mobile phone with the server
R3.2a	The user shall be able to synchronize the contact list on a PDA with the server
R3.2b	The user shall be able to synchronize the calendar on a PDA with the server
R4.1a	The user shall be able to modify a contact item in MRS using a mobile phone
R4.1b	The user shall be able to modify a calendar item in MRS using a mobile phone
R4.2a	The user shall be able to modify a contact item in MRS using a PDA
R4.2b	The user shall be able to modify a calendar item in MRS using a PDA
R4.3a	The user shall be able to modify a contact item in MRS using a computer
R4.3b	The user shall be able to modify a calendar item in MRS using a computer
R5.1a	The user shall be able to add a contact item to the MRS using a mobile phone
R5.1b	The user shall be able to add a calendar item to the MRS using a mobile phone
R5.2a	The user shall be able to add a contact item to the MRS using a PDA
R5.2b	The user shall be able to add a calendar item to the MRS using a PDA
R5.3a	The user shall be able to add a contact item to the MRS using a computer
R5.3b	The user shall be able to add a calendar item to the MRS using a computer
R6.1a	The user shall be able to delete a contact item from the MRS using a mobile phone
R6.1b	The user shall be able to delete a calendar item from the MRS using a mobile phone
R6.2a	The user shall be able to delete a contact item from the MRS using a PDA
R6.2b	The user shall be able to delete a calendar item from the MRS using a PDA
R6.3a	The user shall be able to delete a contact item from the MRS using a computer
R6.3b	The user shall be able to delete a calendar item from the MRS using a computer
R7.1a	The user shall be able to search for a contact item in the MRS using a mobile phone
R7.1b	The user shall be able to search for a calendar item in the MRS using a mobile phone
R7.2a	The user shall be able to search for a contact item in the MRS using a PDA
R7.2b	The user shall be able to search for a calendar item in the MRS using a PDA
R7.3a	The user shall be able to search for a contact item in the MRS using a computer
R7.3b	The user shall be able to search for a calendar item in the MRS using a computer
R8.1a	The user shall be able to save a contact item in MRS locally on his/her mobile phone
R8.1b	The user shall be able to save a calendar item in MRS locally on his/her mobile phone
R8.2a	The user shall be able to save a contact item in MRS locally on his/her PDA
R8.2b	The user shall be able to save a calendar item in MRS locally on his/her PDA
R8.3a	The user shall be able to save a contact item in MRS locally on his/her computer
R8.3b	The user shall be able to save a calendar item in MRS locally on his/her computer
R9	It shall be possible to display only available people, i.e. persons who do not currently have appointments in their calendar (related to requirements R7.**x**a)
R10	The central database must give the possibility to register a contact's role/title and group/department membership with a contact
R11	It shall be possible to schedule group events, i.e. choose one or more other users to include in the appointment
R12	The system shall be able to issue server alerts when items change, requesting users to initiate synchronization

5 System description

This chapter presents the system used in the assignment. First, an overview of the system is given. The synchronization server is described in further detail in the subsequent chapter. How to facilitate the synchronization between mobile devices and the server is also explained. The implementation focus is directed towards user functionality. Hence, this constitutes the major part of this chapter. Both web browser interfaces designed for computers and interfaces for mobile phones (WAP) are described in detail, as well as the underlying functionality. Finally, results from the testing of this functionality, as well as the synchronization testing are given.

5.1 System overview

Figure 5.1 shows a high level view of the system. It consists of a synchronization server (Sync4j server), a Microsoft Exchange server and mobile devices such as mobile phones and PDAs. Through the Sync4j server, the users will synchronize their contact and calendar information on their mobile devices with the data stored in the Exchange server. In this implementation, the database resides on the same machine as the Sync4j server, but this database could might as well be located at a remote location. The Exchange server is located in the same intranet as the Sync4j server, and is also accessible from outside this network. As with the database, the Exchange server could also be located at a remote location.

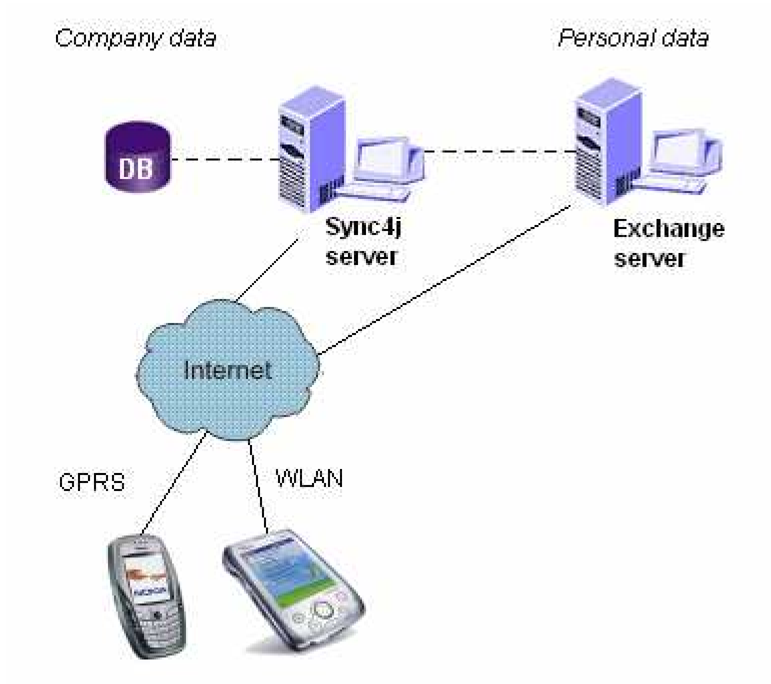


Figure 5.1: System overview
Mobile phones and PDAs connect to the Sync4j server over a GPRS and WLAN connection, respectively. The Sync4j server has a database with central company data, and is also connected to an Exchange server that stores the users' personal data.

The solution presented in this report will be based on storing central company information using the Sync4j server's file system database. The users will also have an Exchange account that can be used to store personal contacts, calendar events etc. The data stored in Exchange is also the data that is synchronized with the users' mobile devices. Users can search for central company data using a web interface (for PDAs and computers) and a WAP interface (for mobile phones). Note that the Exchange server is used in this implementation for storing personal data, but the personal data can be stored in any other database, on the synchronization server or on a remote server. Hence, if the term "Exchange server" is used in a general context in this report, it can also be interpreted simply as an instance hosting the personal data.

5.2 Server

The server used in the implementation is Sync4j bundle 2.2 beta 4 for Windows. The bundle includes the Apache Tomcat 5.0 server. Administration of the server is done with SyncAdmin 1.1.3, also bundled. The Sync4j server will hereafter be referred to as the SyncServer.

5.2.1 SyncConnector Exchange

To connect the SyncServer with the Exchange server, a module is installed in the SyncServer. This module is downloadable from the Sync4j web site [10]. It contains a SyncConnector, SyncConnector Exchange. A SyncConnector is an extension to the SyncServer, integrating it with an external data source [11].

When the SyncServer receives a synchronization request from a client, it will look for updates in its local cache [12]. If some of the items have been modified, it will initiate a WebDAV (Web-based Distributed Authoring and Versioning) connection to the Exchange server. The WebDAV protocol is an extension to HTTP and facilitates the development of writable web applications. This makes it possible to perform operations such as create, copy, delete, move and search for resources in the Exchange datastore.

In the system presented in this report, the user's personal contacts and calendar can be synchronized with his/her personal Exchange account. To make this possible, an Exchange Contact SyncSource and an Exchange Calendar SyncSource must be added in SyncAdmin to integrate the SyncServer with the Exchange datastore. This procedure is explained in detail in Appendix C.

5.2.2 Authentication

For the user to be able to synchronize, he/she must be authenticated. Each user must be added to the list of users in the SyncServer. In addition, the user must be able to synchronize using different devices. All devices must be added to the list of devices in the SyncServer. Finally, a pairing of users and devices must be added. A <user,device> combination is referred to as a principal.

5.2.3 File system database

As mentioned earlier, the SyncServer includes its own file system database for storing data. Each file constitutes a data item and the file is in XML format. Figure 5.2 gives an example of a file representing a contact.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<contact>
  <Email1Address>vbrown@mail.com</Email1Address>
  <BusinessTelephoneNumber>+1 329 5864389</BusinessTelephoneNumber>
  <FirstName>Vincent</FirstName>
  <NickName></NickName>
  <Title></Title>
  <JobTitle>CEO</JobTitle>
  <FileAs>Brown,Vincent</FileAs>
  <MiddleName></MiddleName>
  <Suffix></Suffix>
  <LastName>Brown</LastName>
  <Revision>20050407T150934</Revision>
</contact>
```

Figure 5.2: XML representation of a contact

The contact (Vincent Brown) is stored as a single XML formatted file in a contacts folder. The content of the figure is taken from an example contact file in the Sync4j installation.

5.3 Web user functionality

The system shall offer the possibility to search among the data stored in the central database. The user must be able to perform the search from a set of devices: a computer, a PDA and a mobile phone. User interfaces for these different devices have been implemented.

The user interfaces are implemented only for the file system database, i.e. not for the Exchange system. To search for data stored on the Exchange server, the user can search for contacts and other information using for instance Outlook Web Access.

In the Sync4j bundle there is a web demo included that lists contacts and calendar information, and also gives the possibility to add, delete and modify this information. The web interface is developed using JSP pages and a Java servlet and these are extended to include searching functionality and the option to save vCards and iCalendar representation of the items. Since the original demo only includes a limited set of contact information fields, many fields have been added to the interface. Appendix A further details the servlet implementation and JSP pages. In this appendix, all screenshots relating to the web interfaces are gathered, and the following chapters will include references to these screenshots.

The front menu (see Figure A.5), gives the user the options to search for a contact, add a new contact, search for an event or add a new event.

5.3.1 Search for contacts

The user can search for the following information regarding a contact (see Figure A.6):

- Name
- Company
- Job title
- Department
- Role
- Telephone number
- Address

These fields were chosen from the author's consideration of what kind of information that would be most necessary to be able to search for in an organization. If no information is entered, the search will list all contacts in the database.

Name

A match occurs when all the words entered are substrings of the concatenation of the contact's salutation (e.g. Mr., Mrs., Ms., Dr.), first name, middle name, surname and suffix (e.g. Jr., Sr.).

Company

A match occurs when all the words entered are substrings of the contact's company.

Job title

A match occurs when all the words entered are substrings of the contact's job title.

Role

A match occurs when all the words entered are substrings of the contact's role.

Telephone number

The user can enter a string that is compared to the contact's home phone number, business phone number, cell number and home fax number. The entered number will be cleaned, i.e. any other characters than digits will be removed from the string. This implies that the string entered will be handled as a string of consecutive digits (e.g. +1 (123) 456 78 will become

112345678). The same cleaning procedure is applied to the contact's various phone numbers, and the entered string is compared to each of the contact's phone numbers. If it is a substring of at least one of these phone numbers, a match occurs.

Address

A match occurs when all the words entered are substrings of the concatenation of the contact's work address (street, postal code, city, state and country). The entered search words can also be substrings of the concatenation of the contact's home address (street, postal code, city, state and country).

5.3.2 Add new contact

The user also has the option to add a new contact to the central company database. This option includes more fields than the search option. A partial view of the add contact interface is shown in Figure A.7. The text fields are divided in three departments: General, Personal and Business. The fields are listed in Table 5.1, but further details about them are omitted since they are reasonably self-explanatory.

Table 5.1: Contact fields for add contact interface

General	Personal	Business
• Title (Salutation)	• Street	• Company
• First name	• Postal code	• Department
• Middle name	• City	• Job title
• Last name	• State	• Role
• Suffix	• Country	• Street
• Nickname	• Birthday	• Postal code
• Note	• Home phone	• City
	• Home fax	• State
	• Mobile phone	• Country
		• Business phone
		• E-mail

5.3.3 Contact details

When the user has initiated a search and the results are shown on the screen, he/she can click on the desired contact and view the contact's details. In this view, the user can modify the information stored for this contact and also delete the entire contact listing. A vCard representation of the contact can also be downloaded. Figure A.9 shows a partial view of the details of a sample contact.

5.3.4 Search for events

The users can also search for events. He/she can enter search criteria in the following fields:

- Subject
- Starting date after
- Starting date before
- Event type
- Location
- Description
- Display only future events (checked is standard)

If no information is entered, the search will list all future events in the database.

Subject
A match occurs when all the words entered are a substring of the event's subject.

Starting date after
The user can specify that only events after and including this date will be listed in the search results.

Starting date before
The user can specify that only events prior to and including this date will be listed in the search results.

Event type
A match occurs when all the words entered are a substring of the event's type.

Location
A match occurs when all the words entered are a substring of the event's location.

Description

A match occurs when all the words entered are a substring of the event's description.

Display only future events

If checked, the search will only yield events occurring on the current date or after the current date.

5.3.5 Add new event

The user also has the option to add a new event to the central company database. The add event interface is shown in Figure A.12. The fields are listed below, but further details about them are omitted since they are reasonably self-explanatory.

- Subject
- Starting date
- Starting time
- Ending date
- Ending time
- Event type
- Location
- Description

5.3.6 Event details

When the user has initiated a search and the results are shown on the screen, he/she can click on the desired event and view the event details. In this view, the user can modify the information stored for this event and also delete the entire event listing. An iCalendar representation of the event can also be downloaded. Figure A.13 shows the details of a sample event.

5.4 WAP user functionality

An interface for mobile phones has also been developed using the same skeleton. The JSP pages have been rewritten in WML [13] for this purpose. For usability and to limit the amount of data transferred, this interface is somewhat lighter than the web interface, not including as many text fields.

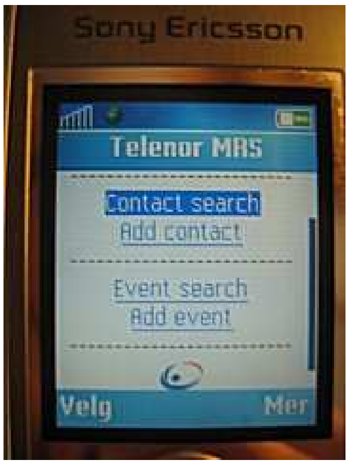


Figure 5.3: WAP front menu

5.4.1 Contacts

The WAP contact interfaces are displayed in Figure 5.4.

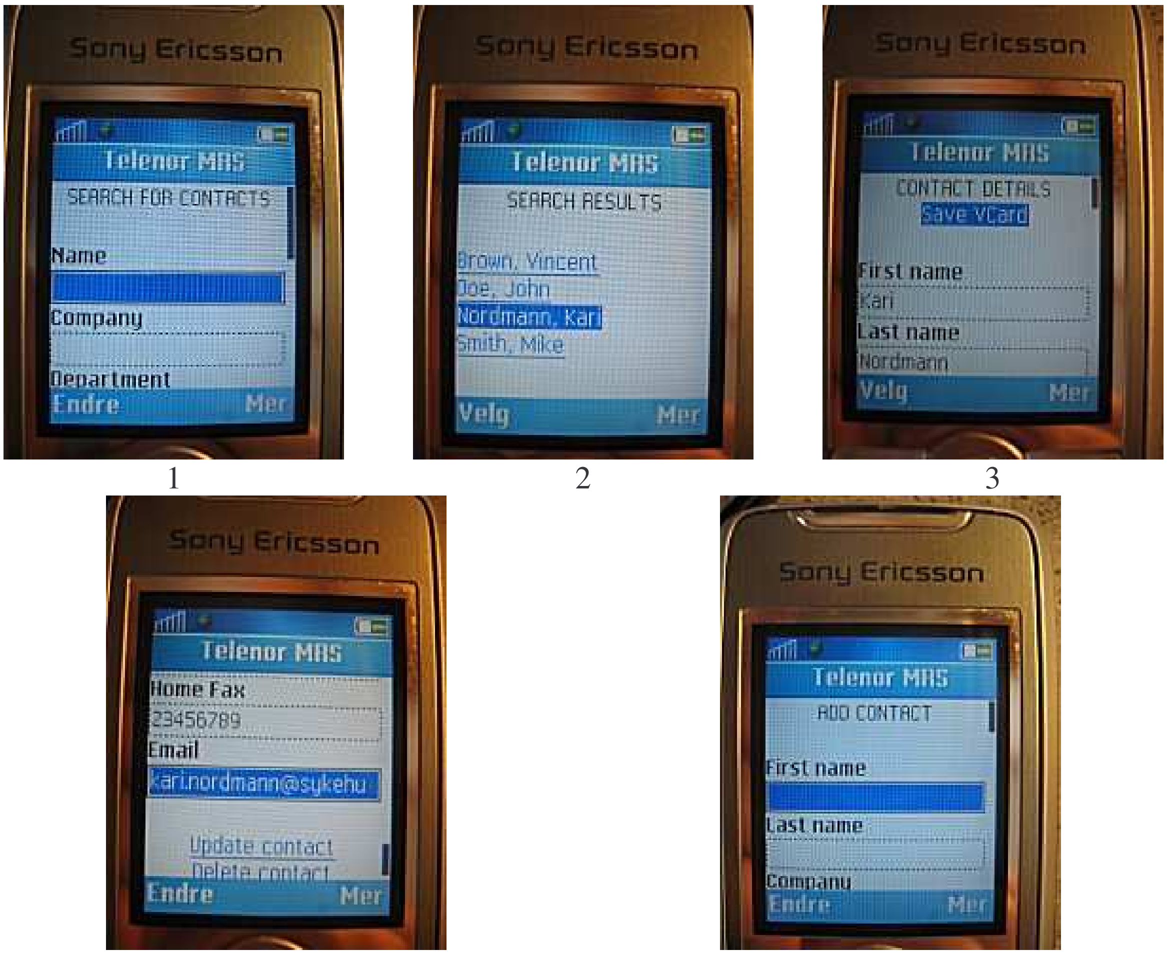


Figure 5.4: Screen photos of contact interfaces
1: Search for contacts. 2: Search results. 3: Contact details (top of page). 4: Contact details (bottom of page). 5: Add contact.

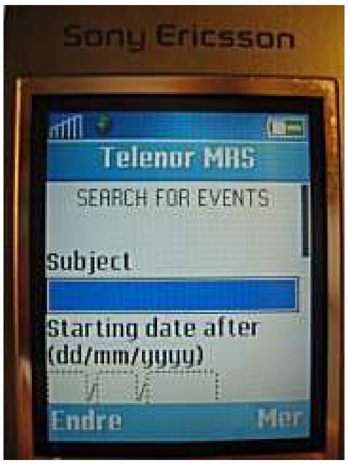
Table 5.2 lists the attributes (input text fields) for the various contact interfaces. The processing and content of these fields are identical to the web interface. For the description of these fields, see chapters 5.3.1-5.3.3.

Table 5.2: Input fields for WAP contact interfaces

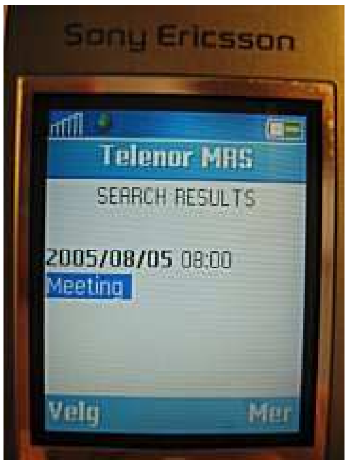
Interface	Input fields
Search for contacts	• Name • Company • Department • Job title • Role • Telephone number • Address
Contact details / Add contact	• First name • Last name • Company • Department • Job title • Role • Work address (street, postal code, city, state, country) • Home address (street, postal code, city, state, country) • Mobile phone • Business phone • Home phone • Home fax • E-mail

5.4.2 Events

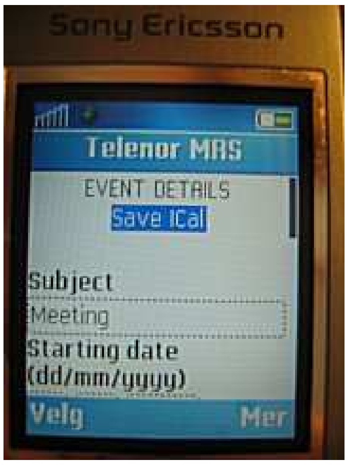
The WAP event interfaces are displayed in Figure 5.5.


Sony Ericsson
Telenor MRS
SEARCH FOR EVENTS
Subject
Starting date after
(dd/mm/yyyy)
/ /
Endre
Mer

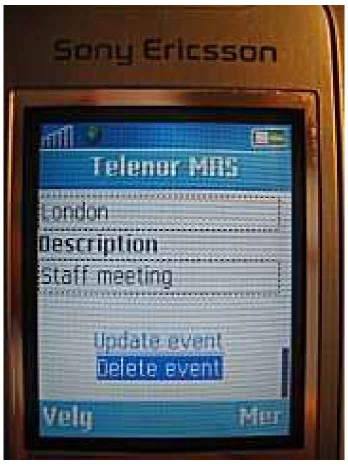
1


Sony Ericsson
Telenor MRS
SEARCH RESULTS
2005/08/05 08:00
Meeting
Velg
Mer

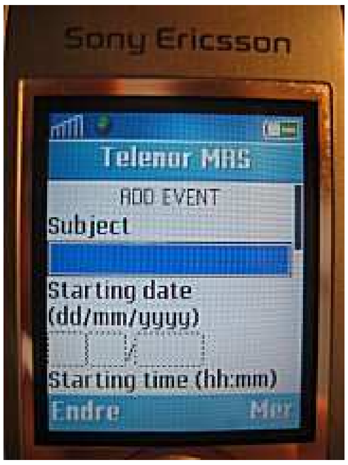
2


Sony Ericsson
Telenor MRS
EVENT DETAILS
Save ICal
Subject
Meeting
Starting date
(dd/mm/yyyy)
Velg
Mer

3


Sony Ericsson
Telenor MRS
London
Description
Staff meeting
Update event
Delete event
Velg
Mer

4


Sony Ericsson
Telenor MRS
ADD EVENT
Subject
Starting date
(dd/mm/yyyy)
/ /
Starting time (hh:mm)
Endre
Mer

5

Figure 5.5: Screen photos of event interfaces

1: Search for events. 2: Search results. 3: Event details (top of page). 4: Event details (bottom of page). 5: Add event.

Table 5.3 lists the attributes (input text fields) for the various event interfaces. The processing and content of these fields are identical to the web interface. For the description of these fields, see chapters 5.3.4-5.3.6.

Table 5.3: Input fields for WAP event interfaces

Interface	Input fields
Search for events	• Subject • Starting date after (dd, mm, yyyy) • Starting date before (dd, mm, yyyy) • Event type • Location • Description • Display only future events
Event details / Add event	• Subject • Starting date (dd, mm, yyyy) • Starting time (hh, mm) • Ending date (dd, mm, yyyy) • Ending time (hh, mm) • Event type • Location • Description

5.5 Testing

This section summarizes the test results for both web and WAP interface, as well as the results from the synchronization testing.

5.5.1 Test environment

The following devices/software were used to verify the implementation:

- Mobile phone: Sony Ericsson k700i
- PDA: HP iPAQ (Microsoft Pocket PC Version 4.20.1081, build 13100), using Internet Explorer for Pocket PC
- Web browser (computer): Mozilla Firefox version 1.0.3

5.5.2 User functionality

Table 5.4 shows the different scenarios tested regarding contacts (search, add, modify, delete, save). The actions listed in the table can be seen as a consecutive series of actions. These actions are grouped, and the groups are each given an identifier, T*x*.

Table 5.4: Contact test scenarios

Action	Expected outcome	Web a	PDA b	Phone c
Click on link "Contact search"	Search page is displayed (Figure A.6)	OK	OK	OK
T1: List all contacts				
Click "Search", with no information entered in the text fields	Search results page is displayed, with all contacts listed (Figure A.8)	OK	OK	OK
Click on a contact listed in search results	The contact details page for this contact is displayed (Figure A.9)	OK	OK	OK
T2: vCard representation				
Click "Save vCard"	Download box for the contact's vCard representation (web) / Contact is downloaded and the user is prompted to add it to the contact list (phone)	OK	Not OK	OK
T3: Search for contacts				
Go to "Contact search", and enter information in relevant fields. Click "Search"	Only contacts that contains the entered data in the corresponding fields are displayed	OK	OK	OK
T4: Modify contact				
View the contact details again, and edit the text in all the fields. Click "Modify"	The contact details are listed again, with all the modified fields displayed correctly.	OK	OK	OK
Go to Contact search, and search for the contact just modified. Click "Search".	Search results page is displayed, with the recently modified contact on the list	OK	OK	OK
Click on the contact	The contact details page is displayed, with the modified information	OK	OK	OK
T5: Delete contact				
Click delete	Confirmation dialog box will appear (web) / Back to menu screen (phone)	OK	OK	OK
Confirm dialog box (web)	Menu screen is displayed	OK	OK	N/A
Go to Contact search, and search for the contact just entered. Click "Search".	Search results page is displayed, with the recently deleted contact not on the list	OK	OK	OK
T6: Add contact				
Click on link "Add contact"	Add contact page is displayed (Figure A.7)	OK	OK	OK
Fill in the desired information (use all fields). Click "Add"	The contact details just entered are displayed	OK	OK	OK
Go to Contact search, and search for the contact just entered. Click "Search"	Search results page is displayed, with the recently added contact on the list	OK	OK	OK

As can be seen from Table 5.4, save vCard did not work as intended with the PDA. The download dialog box was opened, but the only possibility was to save it as a JSP file. The file extension would have to be renamed to .vcf, and then opened to be added to the contact list.

The test scenarios for events are very similar to the contact scenarios. Table 5.5 displays the test results. The actions listed in the table can be seen as a consecutive series of actions. These actions are grouped, and the groups are each given an identifier, T*x*.

Table 5.5: Event test scenarios

Action	**Outcome**	**Web a**	**PDA b**	**Phone c**
Click on link "Event search"	Search page is displayed (Figure A.10)	OK	OK	OK
T7: List events				
Click "Search". Make sure the "Display only future events" is checked	Search results page is displayed, with only future events listed (Figure A.11)	OK	OK	OK
Click on an event listed in search results	The event details page for this event is displayed, with the correct information in all fields (Figure A.13)	OK	OK	OK
Go to "Event search", but this time uncheck "Display only future events"	Search results page is displayed, with all events listed	OK	OK	OK
T8: iCalendar representation				
Click "Save iCal"	Download box for the event's iCal representation (web) / Event is downloaded and the user is prompted to add it to the calendar (phone)	OK	Not OK	OK
T9: Search for events				
Go to "Event search", and enter information in all fields. Click "Search"	Only events that contains the entered data in the corresponding fields are displayed	OK	OK	OK
T10: Modify event				
View the event details again, and edit the text in all the fields. Click "Modify"	The event details are listed again, with all the modified fields displayed correctly.	OK	OK	OK
Go to Event search, and search for the event just modified. Click "Search".	Search results page is displayed, with the recently modified event on the list	OK	OK	OK

Click on the event	The event details page is displayed, with the modified information	OK	OK	OK
T11: Delete event				
Click delete	Confirmation dialog box will appear (web) / Back to menu screen on mobile phone	OK	OK	OK
Confirm dialog box (web)	Menu screen is displayed	OK	OK	N/A
Go to Contact search, and search for the contact just entered. Click "Search".	Search results page is displayed, with the recently deleted contact not on the list	OK	OK	OK
T12: Add event				
Go back to menu, and click on link "Add event"	Add event page is displayed (Figure A.12)	OK	OK	OK
Fill in the desired information (use all fields). Click "Add"	The event details just entered are displayed	OK	OK	OK
Go to Event search, and search for the event just entered. Click "Search"	Search results page is displayed, with the recently added event on the list	OK	OK	OK

As can be seen from Table 5.5, save iCalendar did not work as intended with the PDA. The download dialog box was opened, but the only possibility was to save it as a JSP file. The file extension would have to be renamed to .ics, and then opened to be added to the calendar.

5.5.3 Synchronization with Exchange server

Table 5.6 shows the different scenarios tested regarding contact and event synchronization. The actions listed in the table are grouped, and the groups are each given an identifier, T*x*. Outlook Web Access was used to modify/add/delete items on the server.

Table 5.6: Synchronization test scenarios

Action	Outcome	PDA a	Phone b
T13: Transfer of contacts from server			
Start out with an empty contact list on the device, and synchronize with the contacts on the server	All the items on the server displayed in the device's contact list	OK	OK
T14: Modify contact			
Modify a contact on the device and synchronize with the server	The modified contact is also modified on the server	OK	OK
Modify a contact on the server and initiate synchronization from the device	The modified contact is also modified on the device	OK	OK
T15: Add contact			

Add a new contact on the device and synchronize with the server	The new contact is also added on the server	OK	OK
Add a new contact on the server and initiate synchronization from the device	The new contact is also added on the device	OK	OK
T16: Delete contact			
Delete a contact on the device and synchronize with the server	The contact is also deleted from the server	OK	OK
Delete a contact on the server and initiate synchronization from the device	The contact is also deleted from the device	OK	OK
T17: Transfer of events from server			
Start out with an empty calendar on the device, and synchronize with the events on the server	All the items on the server displayed in the device's calendar	OK	OK
T18: Modify event			
Modify an event on the device and synchronize with the server	The modified event is also modified on the server	OK	OK
Modify an event on the server and initiate synchronization from the device	The modified event is also modified on the device	OK	OK
T19: Add event			
Add an event on the device and synchronize with the server	The new event is also added on the server	OK	OK
Add an event on the server and initiate synchronization from the device	The new event is also added on the device	OK	OK
T20: Delete event			
Delete an event on the device and synchronize with the server	The event is also deleted from the server	OK	OK
Delete an event on the server and initiate synchronization from the device	The event is also deleted from the device	OK	OK

The PDA tests for deleting items (T16 and T20) occasionally experienced read timeouts when synchronizing, and the PDA had to be reset. This was quite annoying, and the reason for this error is unknown. This never occurred with the mobile phone, but quite frequently on the PDA, so it is reason to believe that the problem is related to the SyncClient PIM Pocket PC client.

Another observation from the testing is related to displaying contacts' names. When adding a new item on the mobile phone, e.g. Hans Olsen, the name shown in Outlook Web Access was on the format Hans, Olsen, i.e. the first and last name has been switched. On the other hand, if the contact's details are viewed, the first and last names are correct. Hence, this is a problem with the display name.

Also, the mobile phone used does not separate between first name and last name. People with two or more first names, e.g. Hans Erik Olsen, will be added with first name "Hans" and last name "Erik Olsen" in Exchange. The mobile phone gives the possibility to set the synchronization sequence to last name (first name is default), so that the contacts are saved as "last name, first name" on the phone (default "first name last name"). This way, you would enter a new contact as "Olsen, Hans Erik", and then the correct first and last names are stored in Exchange as well.

A final note is that only the contact's home address as entered in Outlook Web Access is displayed on the phone.

5.5.4 Summary of test results

Table 5.7 summarizes the test results. The requirements worked out in chapter 4.2 are divided into contacts and events, and the table shows whether the requirement is fulfilled for the different devices. Finally, the corresponding test scenario according to the previous chapters is given.

Table 5.7: Summary of test results

ID	Description	Status	Test scenario
R1	The system shall use an open source implementation of a server	OK (Sync4j)	
R2	The system shall use SyncML DS over GPRS and/or WLAN	OK	
R3.1a	The user shall be able to synchronize the contact list on a mobile phone with the server	OK	T13b-T16b
R3.1b	The user shall be able to synchronize the calendar on a mobile phone with the server	OK	T17b-T20b
R3.2a	The user shall be able to synchronize the contact list on a PDA with the server	OK	T13a-T16a
R3.2b	The user shall be able to synchronize the calendar on a PDA with the server	OK	T17b-T20b
R4.1a	The user shall be able to modify a contact item in MRS using a mobile phone	OK	T4c
R4.1b	The user shall be able to modify a calendar item in MRS using a mobile phone	OK	T10c
R4.2a	The user shall be able to modify a contact item in MRS using a PDA	OK	T4b
R4.2b	The user shall be able to modify a calendar item in MRS using a PDA	OK	T10b
R4.3a	The user shall be able to modify a contact item in MRS using a computer	OK	T4a
R4.3b	The user shall be able to modify a calendar item in MRS using a computer	OK	T10a
R5.1a	The user shall be able to add a contact item to the MRS using a mobile phone	OK	T6c
R5.1b	The user shall be able to add a calendar item to the MRS using a mobile phone	OK	T12c
R5.2a	The user shall be able to add a contact item to the MRS using a PDA	OK	T6b
R5.2b	The user shall be able to add a calendar item to the MRS using a PDA	OK	T12b
R5.3a	The user shall be able to add a contact item to the MRS using a computer	OK	T6a

R5.3b	The user shall be able to add a calendar item to the MRS using a computer	OK	T12a
R6.1a	The user shall be able to delete a contact item from the MRS using a mobile phone	OK	T5c
R6.1b	The user shall be able to delete a calendar item from the MRS using a mobile phone	OK	T11c
R6.2a	The user shall be able to delete a contact item from the MRS using a PDA	OK	T5b
R6.2b	The user shall be able to delete a calendar item from the MRS using a PDA	OK	T11b
R6.3a	The user shall be able to delete a contact item from the MRS using a computer	OK	T5a
R6.3b	The user shall be able to delete a calendar item from the MRS using a computer	OK	T11a
R7.1a	The user shall be able to search for a contact item in the MRS using a mobile phone	OK	T1c/T3c
R7.1b	The user shall be able to search for a calendar item in the MRS using a mobile phone	OK	T7c/T9c
R7.2a	The user shall be able to search for a contact item in the MRS using a PDA	OK	T1b/T3b
R7.2b	The user shall be able to search for a calendar item in the MRS using a PDA	OK	T7b/T9b
R7.3a	The user shall be able to search for a contact item in the MRS using a computer	OK	T1a/T3a
R7.3b	The user shall be able to search for a calendar item in the MRS using a computer	OK	T7a/T9a
R8.1a	The user shall be able to save a contact item in MRS locally on his/her mobile phone	OK	T2c
R8.1b	The user shall be able to save a calendar item in MRS locally on his/her mobile phone	OK	T8c
R8.2a	The user shall be able to save a contact item in MRS locally on his/her PDA	Not as intended	T2b
R8.2b	The user shall be able to save a calendar item in MRS locally on his/her PDA	Not as intended	T8b
R8.3a	The user shall be able to save a contact item in MRS locally on his/her computer	OK	T2a
R8.3b	The user shall be able to save a calendar item in MRS locally on his/her computer	OK	T8a
R9	It shall be possible to display only available people, i.e. persons who do not currently have appointments in their calendar (related to requirements R7.**x**a)	Not implemented	-
R10	The central database must give the possibility to register a contact's role/title and group/department membership with a contact	OK	T6
R11	It shall be possible to schedule group events, i.e. choose one or more other users to include in the appointment	Not implemented	-
R12	The system shall be able to issue server alerts when items change, requesting users to initiate synchronization	Not implemented	-

Note regarding R12: Although server alerted synchronization is specified in the SyncML protocol, see chapter 3.1.2, the most recent Sync4j server implementation does not offer this possibility [1]. Therefore, the users have to initiate synchronization themselves, e.g. in the beginning of their work shift and possibly at regular intervals throughout the shift.

6 Business considerations

This chapter will look into which roles are needed if the Mobile Repository System shall be deployed in the market. Business relationships between these roles will also be discussed, and three scenarios for deploying the service are investigated – internal network, internal service and public service. These scenarios are accompanied by an illustrating case.

6.1 Roles

If the Mobile Repository System shall be deployed in the market, there are many actors needed to realize the service. These actors need to cover certain roles, as described below.

End user
The end user is the person who operates the mobile device and synchronizes its data with the server datastore.

Customer
The customer role is the instance that pays for the service in question. The payment could consist of the charge of the transportation and storage of data, subscription fee etc.

Connectivity provider
The connectivity providers will provide the users with Internet access. This could be an ISP or a mobile phone operator offering Internet access through e.g. GPRS.

Synchronization provider
The synchronizing instance is typically the synchronization server. In the case of Mobile Repository System, this is the Sync4j SyncServer. This server could be installed at and operated by the customer, or it could be left to a third party.

Datastore provider
The datastore provider facilitates the storage of the central company data relevant to synchronization. The datastore could reside along with the synchronization server, or it could be a separate instance stored in a remote location. The datastore provider can therefore be the

same as the synchronization provider, either the customer or a third party. The synchronization provider and datastore provider could also be two separate parties, with the synchronization provider buying storage from an external datastore provider.

Retailer

The retailer is the role that sells the actual service to the customer. The retailer role can be decomposed in several different roles, as depicted in Figure 6.1.

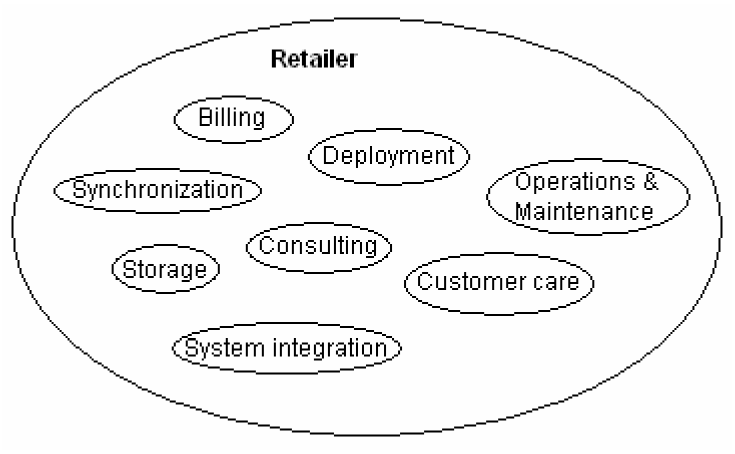


Figure 6.1: Role decomposition of the retailer (modified from [14])

The retailer should provide consulting services to assess the company's needs and to come up with the appropriate solution. The roles indicate that the retailer can be hosting the synchronization server and store the customer's data. No matter which instance is hosting the service, the retailer will have a key role in the deployment, or installation, of the service. System integration might be needed for integrating the service with the customer's current system and/or if the complexity of the synchronization and datastore scheme requires special integration efforts. Operations and maintenance is needed to make sure that the service is functioning properly at all times. Customer care will try to acquire new customers and follow-up the existing customers to make sure that they are satisfied. Billing (subscriptions, payment per use) is also a key role.

6.2 Business relationships

Figure 6.2 shows the business relationships between the participating roles. The customer buys a service from a retailer. The retailer can have agreements with a third party synchronization provider and datastore provider, but the retailer could also provide these services. The customer, retailer, synchronization provider and datastore provider all have business relationships with a connectivity provider for Internet access and the transportation of data. The different connectivity providers involved must also have relationships between each other. Finally, the relationship between customer and end user could be for example employee-employer.

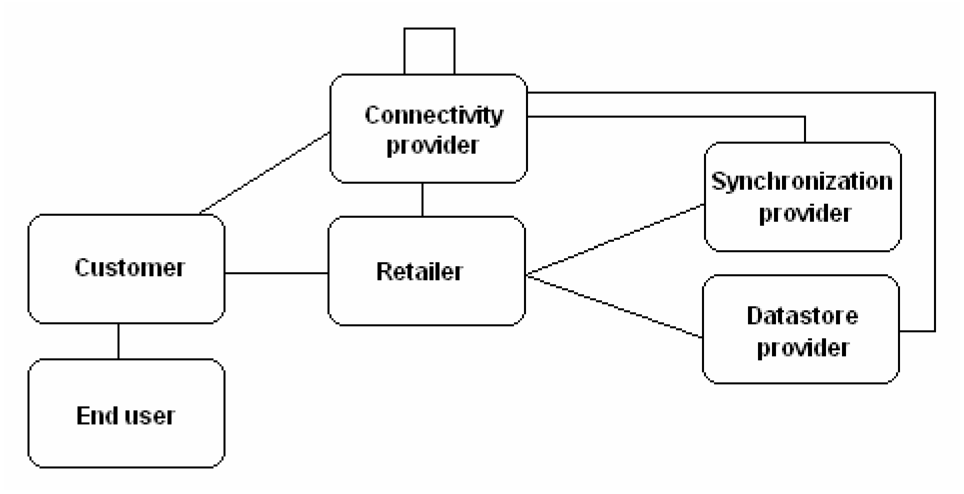


Figure 6.2: Business relationships
The lines illustrate business relationships between the roles involved.

From the figure, the synchronization provider and datastore provider are separate roles. These roles might instead be covered as a part of the retailer's roles, as indicated in Figure 6.1.

6.3 Service deployment

Three different service deployment scenarios will be described next – internal network, internal service and public service. The business relationships displayed in Figure 6.2 are valid for all these scenarios, but the roles of an actor in one scenario might be different from the actor's roles in another scenario. Especially, the customer role and the retailer role are subject to change.

6.3.1 Internal network

In this case, the synchronization server and optionally the storing of data are located in the internal network, see Figure 6.3. This could be beneficial for relatively large companies. A retailer can offer a customized solution to be installed as a part of the company's internal network. The synchronization server must be accessible from outside the internal network.

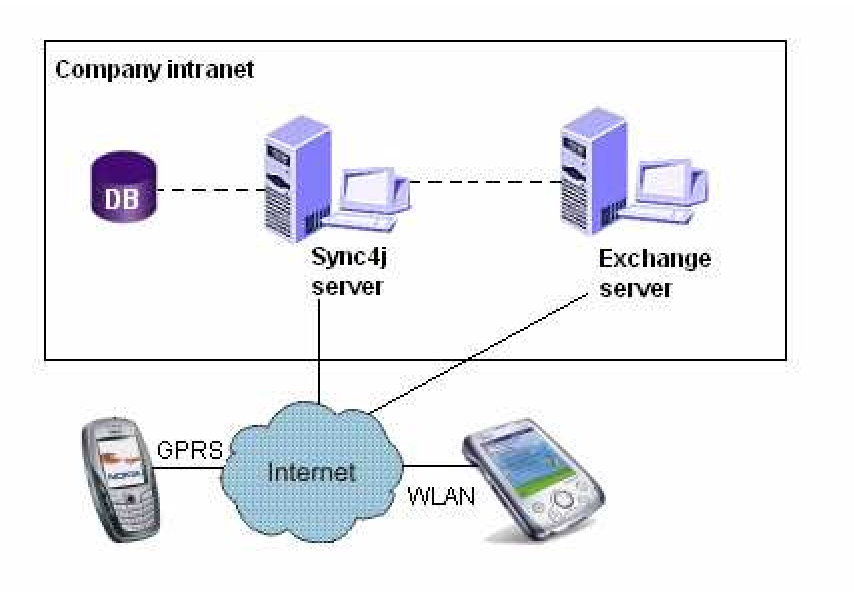


Figure 6.3: Service realized in an internal network

Possible role decompositions of the retailer and customer are drawn in Figure 6.4.

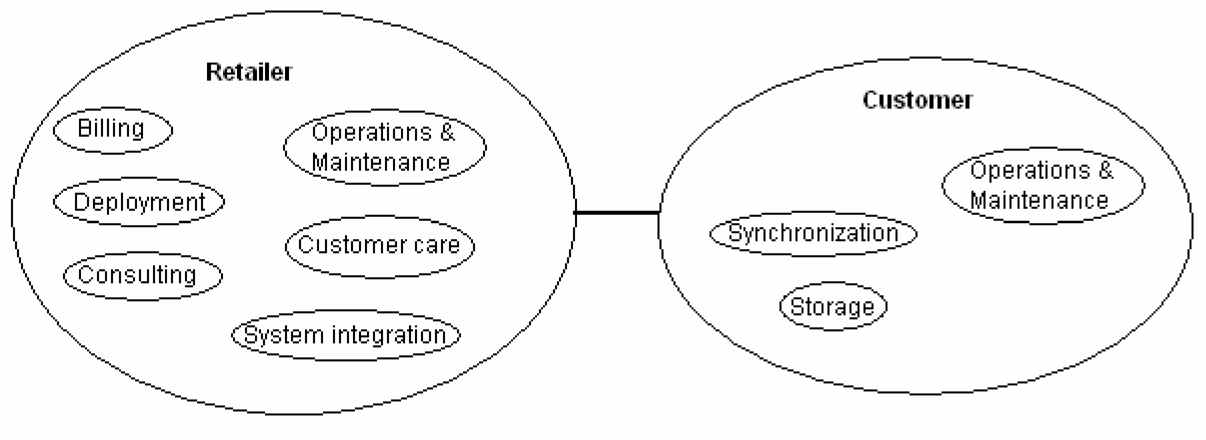


Figure 6.4: Retailer and customer roles for the internal network scenario

What makes Figure 6.4 different from Figure 6.3 is that the customer now is hosting the synchronization and datastore server. Otherwise, the retailer will assist the customer in all the aspects previously explained. The customer can perform some operations and maintenance relating to the service, so this is considered to be an overlapping role.

Case illustration

St. Olav's Hospital in Trondheim is going to install MRS. Due to its size, it is natural that the hospital itself hosts the synchronization server and stores the data. Thus, the hospital has the role as the customer, as well as the synchronization provider and datastore provider. Telenor Business Solutions, as the retailer, could assist the hospital with the deployment of MRS and integrating it with the hospital's current system. Telenor can also function as the hospital's connectivity provider, both as an ISP and as a mobile phone operator offering GPRS access (Telenor Mobil). Hence, revenue is generated by offering connectivity and also a compensation for the deployment of MRS. In addition, Telenor could offer e.g. upgrades and operations/maintenance, either at a fixed price or as a subscription.

The actor/role mappings are listed below. The role decompositions shown in Figure 6.4 still apply.

End user	Employer (nurses, doctors, administrative personnel etc.)
Customer	St. Olav's Hospital
Connectivity provider	Telenor / Telenor Mobil
Synchronization provider	St. Olav's Hospital
Datastore provider	St. Olav's Hospital
Retailer	Telenor Business Solutions

6.3.2 Internal service

In this scenario, the customer purchases the service from a retailer. The retailer itself can host the synchronization server and datastore, or it could make use of third party service providers for these purposes. The retailer and/or the third party service provider are responsible for the operations and maintenance of the service.

Figure 6.5 uses the scenario with the third party service provider as an example. Note that all the roles associated with the third party could be the sole responsibility of the retailer.

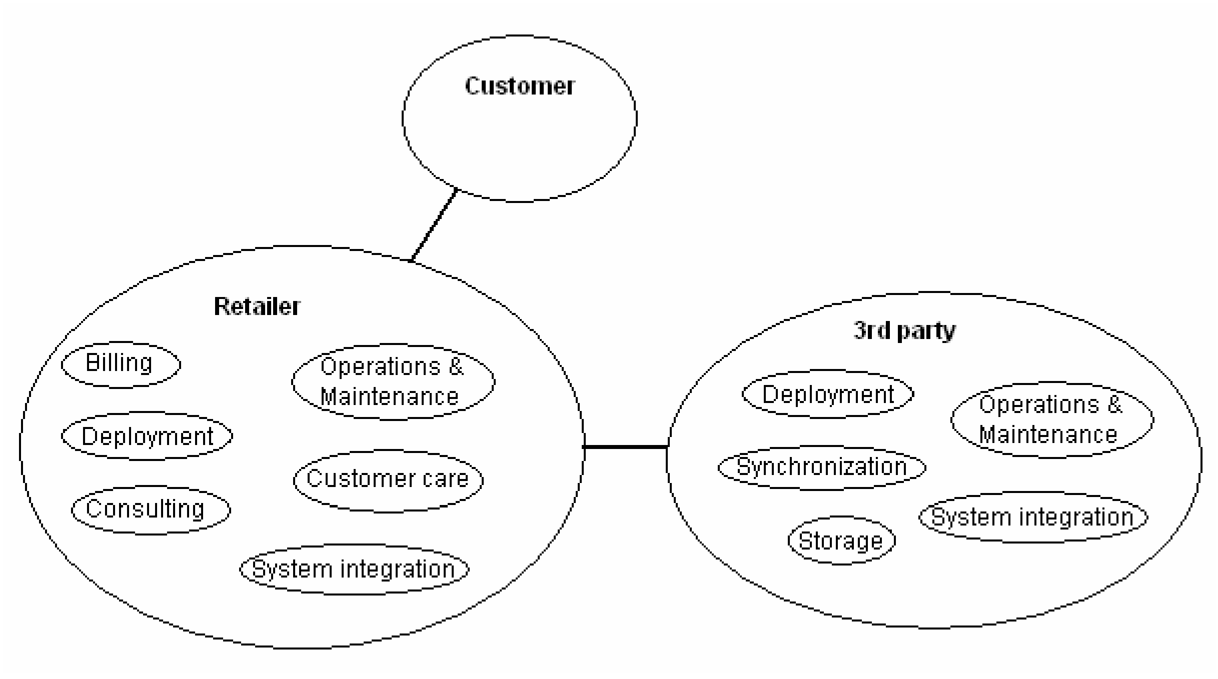


Figure 6.5: Retailer, customer and third party roles for the internal service scenario

Now, the customer is not directly involved in hosting and operations/maintenance, and the customer is only included in the figure to illustrate its business relationship with the retailer. Note that it is the *retailer* that has a business relationship with the third party service provider, the customer only has an indirect relationship through the retailer.

There are a number of overlapping roles in this scenario. The third party service provider most likely plays an active role in the deployment and system integration phases. It would also be necessary that the third party contributes to operations and maintenance. The roles of billing and customer care are meant to be associated with the customer that buys the service, so these roles are located at the retailer, since the retailer is the customer's point of contact. Consulting could also be an overlapping role between the retailer and the third party service provider, but this possibility is not illustrated in the figure.

Case illustration

A medium sized company wants to install MRS. It does not have the capacity to run the synchronization server itself, and therefore wants to outsource it. This company has specific needs, and Telenor Business Solutions, as the retailer, is contacted to help developing a solution that will make sure the company's requirements are fulfilled. In this case, Telenor will be the synchronization provider and possibly also the datastore provider. Telenor can also function as the company's connectivity provider, both as an ISP and as a mobile phone operator offering GPRS access (Telenor Mobil). Telenor could require a monthly subscription fee for offering this service to the company. In addition, revenue is generated by offering connectivity. Finally, Telenor could offer e.g. upgrades and operations/maintenance, either at a fixed price or as a part of the subscription.

The actor/role mappings are listed below. The role decompositions shown in Figure 6.5 still apply, except that in this case, Telenor does not make use of a third party. Thus, the Retailer and the 3rd Party in the figure can be merged together.

End user	Employer
Customer	Company
Connectivity provider	Telenor / Telenor Mobil
Synchronization provider	Telenor Business Solutions
Datastore provider	Telenor Business Solutions
Retailer	Telenor Business Solutions

6.3.3 Public service

For the least demanding customers, the public service could be satisfactory. The customers, either small companies or individuals, purchase or subscribe to standard, non-customizable solutions from a retailer. The retailer itself can host the synchronization server and datastore, or it could make use of third party service providers for these purposes.

Figure 6.6 illustrates the roles associated with the retailer and the customer. As for the internal service, the customer does not have any specific roles, but is included in the figure to indicate its business relationship with the retailer.

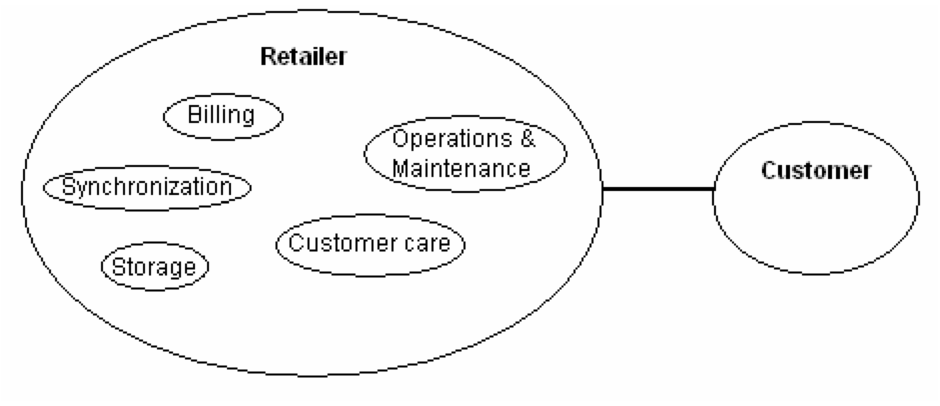


Figure 6.6: Retailer and customer roles for the public service scenario

First, note that Figure 6.6 illustrates an example of the retailer providing all the services itself, without the use of a third party service provider. A third party could of course also be involved, as for the internal service scenario.

As can be seen from the figure, the retailer is somewhat 'lighter' than previously illustrated. The system integration, deployment and consulting roles are all removed. This is due to the fact that the public service is more standardized and not as customizable. The retailer offers a package featuring synchronization and storage for use with specific software and systems, supporting certain devices, but if the customer has needs beyond what is included and supported in the public service, it must choose either the internal network or the internal service solution.

Case illustration

John Doe would like to have access to his contacts and calendar information from any location using different devices. In case he loses his mobile phone, this solution would also function as a backup of his data. Telenor, as the retailer, offers a standard solution directed towards individuals and small businesses that offer this functionality. Telenor will be the synchronization provider and the datastore provider, and it can also be John Doe's connectivity provider, both as an ISP and as a mobile phone operator offering GPRS access (Telenor Mobil). Telenor requires a monthly subscription fee for offering this service to Mr. Doe. In addition, revenue is generated by offering connectivity.

The actor/role mappings are listed below. The role decompositions shown in Figure 6.6 still apply.

End user	John Doe
Customer	John Doe
Connectivity provider	Telenor / Telenor Mobil
Synchronization provider	Telenor
Datastore provider	Telenor
Retailer	Telenor

In this case, one could also imagine that the datastore provider was a 3rd party, like Hotmail, Yahoo or similar. Then, Telenor could for example offer a synchronization option with these 3rd party datastore providers.

7 Discussion

This chapter discusses the current solution with respect to architecture, scalability, reliability, performance, openness and system-interface considerations. Finally, some suggestions for future work are pointed out.

7.1 Architecture

The solution presented makes use of the file system database included with the Sync4j SyncServer for storing the organization's central data. Each item is stored in a separate file in a folder, e.g. contacts/ and calendar/. The users synchronize their mobile devices with the data in their personal Exchange account. Although the data resides on the Exchange server, the synchronization is performed via the SyncServer.

The reason why this architecture was chosen was mainly due to the problem of separating personal and business data on the users' devices. A user is most likely to have both personal contacts and business contacts. When synchronizing, the personal contacts must not be added to the central company datastore. So unless the users have separate devices for work and private use, or the devices themselves are able to distinguish between personal and business data, this problem would persist if all data is synchronized with a common datastore. This is the reason why the users only synchronize with their private Exchange contacts/calendar folder in the solution presented in this report. These folders could contain a mix of personal and business contacts/events. The data in MRS are accessible through web interfaces, and the user can save items in MRS to his/her device, and synchronize with the Exchange server. This scheme most likely leads to consistency problems. If an item is changed in MRS, then the users that have previously saved a copy of this item is not aware of the change.

The original idea was to create public Exchange folders containing contacts, calendar information etc., but SyncConnector Exchange can only synchronize the users' private folders. A workaround to this problem could be to create a generic, common user for synchronization purposes, alternatively a user for each department/group that would synchronize the same data. Still, this would not have solved the problem with separating business and personal data.

It would be useful if the user could synchronize with a subset of the data stored in MRS, e.g. according to which group(s) and department(s) he/she is a part of. One solution could be to make a folder for each group, but then you would have a consistency issue when an item belongs to more than one group. Imagine a user who is a member of group A modifies an item and synchronizes with the server. Then, the item is correct for group A, but if the item is also stored in the group B folder, then this item would not be updated. Therefore, a mapping of elements between folders is needed. This solution would also waste space in storing the item in several folders. On the other hand, the files are very small and disk space is not the major issue. Another problem with having several folders is related to authentication. Basically, a user of the Sync4j SyncServer has access to all the folders if he/she knows the folder name.

Another solution could be to create a database with users and their respective group membership. Each item would have to be assigned an additional data field describing which groups that will synchronize this item. In this way, each item would only be stored once and the consistency and waste-of-space problems would be eliminated. But the problem relating to separation of personal and business data would still remain.

Another approach is to apply a filter to the datastore, specifying which items to synchronize, e.g. all contacts belonging to a certain group. This would solve the problems pointed out earlier in this chapter. Each item only needs to be stored in one location, and this filtering should be specified so that only central company data would be synchronized, excluding personal data. For synchronizing personal data, another datastore could be used, such as e.g. the Exchange server. Unfortunately, such filtering is not currently supported in the Sync4j implementation. Filtering can also be used to improve scalability, and this is discussed further in the next chapter.

It is an easy task to set up synchronization with the data in MRS instead of the Exchange server (see Appendix B), but due to the problem with separating personal and business data, this was not considered to be a good solution.

7.2 Scalability

For large organizations, it is important to be able to divide e.g. the contacts database into groups/departments, and let the users synchronize only with a subset of the entire database. It is not a practical approach to have the users synchronize with the entire database if it consists of several thousand contacts. The same thing goes for calendar information, e.g. synchronize with only events that occur in the week to follow, excluding all later events and all events that have occurred in the past. SyncML specifies a CGI script search filter [5] that can be appended to the target LocURI to perform selection filtering. Unfortunately, this is not implemented in the current release of Sync4j. CGI scripting could be one approach to synchronize only a subset of the data, e.g. just choosing today's calendar events, excluding all earlier/later events, contacts, notes, tasks and other data. An example is illustrated in Figure 7.1.

```
<Target
<LocURI>./calendar?DTSTART&GE;20050601T000000&AND;DTEND&LT;20050608T000000</LocURI>
</Target>
```

Figure 7.1: CGI target address filtering (based on [5])

In this example, calendar database events (located in the folder ./calendar) are filtered so that just events that start on or after June 1, 2005 and end no later than June 8, 2005 are returned[2].

A typical file created in the file system database has a size of a few hundred bytes up to a couple of kilobytes. Thus, storing contact and calendar information would not require much disk space, even in a large organization with a considerable amount of external contacts and employees.

In addition to the amount of data associated with synchronization, the frequency of synchronizations will affect scalability. The frequency of changes to the datastore and the network traffic associated with propagating these changes to the relevant users will influence how often synchronization should be performed. It would generate a lot of network traffic if there are a substantial amount of users that synchronize every time there is a minor change to an item in the datastore. The users will be able to initiate synchronization by themselves as they wish, but automated schemes for synchronizing at regular intervals could be a desired solution. This report only focuses on manual, client initiated synchronization.

[2] &GE; greater than or equal to / &LT; less than

In this solution, there is only one instance of the SyncServer, and this could quickly become a bottleneck in a system with a great number of concurrent users. This is also a reliability issue, and the reader is referred to the next chapter.

7.3 Reliability

In the current installation, Sync4j is running on a single server, and the organization's central data is stored on this server. This is therefore a many-to-one topology, as described in chapter 2.6.2, and the server might be a system bottleneck. Additionally, if the server encounters a failure and the data is lost, there is no way to retrieve it. For redundancy purposes, there must be one ore more additional servers that store a copy of the database, for example topologically organized as a cluster of servers which replicate between each other (see chapter 2.6.4). For increased reliability, these servers should be geographically distributed in case of a network failure.

As for the Exchange server the same problem exists, but in this case the problem is not as critical. The Exchange server stores the users' private contacts and calendar, and in case of a server failure, the users will most likely have the items stored on their mobile devices as well. Hence, they would only need to synchronize a device with the server once it is up and running again, retrieving the lost data.

7.4 Performance

Synchronization time is related to the network connection and amount of data to be synchronized. Time requirements vary according to user needs. For instance, a mobile device can synchronize when a person arrives at work in the morning, and he/she can do something else while waiting for the synchronization to finish. But there may be other uses that involve stricter requirements to synchronization time. Synchronizing only a subset of the data, as discussed in chapter 7.2, will also have positive impact on the synchronization time.

The SyncServer has an option to log all events relating to a synchronization attempt. With the Log all-option, detailed logs containing time, operations and the content of the SyncML messages are displayed. This log file could be used to generate useful data on the synchronization time, content length of messages etc. This type of data can be very useful when Service Level Agreements (SLAs) are worked out. A SLA can contain requirements relating to for example response time and synchronization time related to amount of data.

It could also be interesting to observe the average amount of data sent per synchronization. As long as the pricing scheme is not fixed, this can be used for predicting revenue possibilities.

The log file had information regarding synchronization time consumption during the system testing. Typically, 1-3 modified items were transferred. Table 7.1 lists the values observed for setup time and sync time for the two devices.

Table 7.1: Synchronization time consumption
Sync time could be read from the log file, by checking first and last timestamp for the ongoing session. Total time was calculated by observing the time passed from the moment the user started the synchronization, until the screen displayed that synchronization was completed. Setup time is then the difference between total time and sync time.

Device	Setup time	Sync time	Total time
Mobile phone	4-5 seconds	5-6 seconds	9-11 seconds
PDA	6 seconds	10 seconds	16 seconds

As the table shows, the duration time for synchronizing the PDA is higher than for the mobile phone. The PDA used a WLAN connection and the mobile phone connected with GPRS. The difference in synchronization time consumption could be due to the fact that the mobile phone sent the data in plain, unencoded vCard/iCalendar format, while the PDA used XML for formatting the data, and the data was sent encoded. This is a consequence of how the Exchange SyncSource is implemented, and the data had to be formatted in this way for the respective devices.

7.5 Openness

The Sync4j synchronization server uses SyncML, an open standard for data synchronization. Any device supporting the SyncML protocol will be able to synchronize its data with the system. This standard is widely adopted by mobile phones. Client software for synchronizing with Microsoft Outlook also exists, as well as clients for Pocket PC / Windows Mobile and BlackBerry devices [10]. Data from the central repository can be accessed through interfaces for mobile phones, PDAs and computers, using HTTP.

7.6 System-interface considerations

If the SyncServer is to be installed in a company's internal network, then it must be opened for users outside of the internal network for it to be useful. A user should be able to synchronize with the server and access the Mobile Repository System at any location and at any time. This on the other hand requires robust security mechanisms.

Currently, anyone who knows the address of the SyncServer can search for information in the MRS, as well as add, modify and delete contacts and events. Hence, it is crucial that some form of authentication mechanism is added. If only users registered in the Sync4j SyncServer are allowed to access the MRS, then it would be preferable to link this with the user database for Sync4j. Then, the user logs on to the MRS with the same username and password as he/she uses for synchronization.

Synchronizing with the Exchange server requires that the Exchange user is added to the SyncServer, and that there exists a mapping between the user and the device he/she is using (a so-called principal, as explained in chapter 5.2.2). The SyncServer uses basic authentication to verify the user's credentials, i.e. a Base64 encoded representation of *username:password* is transmitted [5]. In addition, the SyncML protocol requires support for MD5 digest access authentication. Then the server issues a nonce and calculates a hash of the concatenation *username:password:nonce*. This authentication scheme is more secure, as the nonce will change the authentication hash in each message, preventing replay attacks. Currently, MD5 authentication is not supported in the Sync4j implementation.

For improved security, it could also be a good idea to use HTTPS as a transport protocol instead of HTTP.

7.7 Future work

There are many issues mentioned in this and previous chapters that need to be resolved. To summarize, the following list points out areas that could be useful:

- Authentication of users accessing the MRS using web or WAP interfaces

- Extension of the SyncServer to support server alert. Not all devices support server alert (push messages) anyway, so an alternative solution for notifying these devices would be useful, e.g. a messaging system
- Modify Exchange SyncSource to support the synchronization of public folders
- Add support for more data types in MRS, such as notes and tasks, and also documents and files in general
- Implement CGI scripting (filtering) in Sync4j

8 Conclusion

A system consisting of a synchronization server, Sync4j SyncServer, and an Exchange server for storing personal data has been set up. The synchronization server also functions as the central repository for company related data. In addition, functionality for adding, modifying, deleting and searching for items in this repository has been implemented, for computers, PDAs and mobile phones. This system is net-centric and facilitates location independent access and synchronization, and therefore the major goal behind this thesis has been fulfilled.

A problem regarding the solution is the fact that it is not possible to synchronize with a subset of the repository. For large organizations, this is not an acceptable solution and it would lead to a considerable amount of data to be transferred when synchronizing, especially the first time. It would also be cumbersome to handle this amount of data on e.g. a mobile phone. Filtering is a possible solution to this problem, but it is currently not implemented in Sync4j.

The SyncServer is fully accessible from outside the internal network where it is installed, to offer true location independency. This places strict requirements to authentication. Users who synchronize their data are authenticated using basic authentication, but it would be more secure to use MD5 digest access authentication. This is not implemented in Sync4j. The central repository is accessible by anyone who knows the server address, and this is of course not satisfactory in a real solution offered by a retailer.

The solution makes it possible to synchronize contacts and events using the SyncML protocol, as well as perform operations on a central repository also consisting of contacts and events. As mentioned, there are a number of issues needed to be worked out in order to offer a service, but this solution can be seen as a starting point for further development.

References

[1] The Sync4j Project, *Sync4j SyncServer Developer's Guide*, January 2005, http://download.forge.objectweb.org/sync4j/sync4j_syncserver_developer_guide-2.2.pdf

[2] U. Hansmann, R. Mettälä, A. Purakayastha, P. Thompson, *SyncML: Synchronizing and Managing Your Mobile Data*, Upper Saddle River, NJ: Prentice Hall, 2003. ISBN 0-13-009369-6

[3] Open Mobile Alliance, *OMA Technical Section – Affiliates – SyncML,* http://www.openmobilealliance.org/tech/affiliates/syncml/syncmlindex.html

[4] The SyncML Initiative, *SyncML Sync Protocol, version 1.0.1*, 2001, http://www.openmobilealliance.org/tech/affiliates/syncml/syncml_protocol_v101_2001 0615.pdf

[5] The SyncML Initiative, *SyncML Representation Protocol, version 1.0.1,* 2001, http://www.openmobilealliance.org/tech/affiliates/syncml/syncml_represent_v101_2001 0615.pdf

[6] The Sync4j Project home page, http://sync4j.funambol.com

[7] F. Dawson, T. Howes: *vCard MIME Directory Profile,* RFC 2426, September 1998, http://www.ietf.org/rfc/rfc2426.txt

[8] versit Consortium, *vCard: The Electronic Business Card*, Version 2.1, September 18, 1996, http://www.imc.org/pdi/vcard-21.txt

[9] F. Dawson, D. Stenerson, *Internet Calendaring and Scheduling Core Object Specification*, RFC 2445, November 1998, http://www.ietf.org/rfc/rfc2445.txt

[10] The Sync4j Project download page, http://sync4j.funambol.com/main.jsp?main=download_stable

[11] The Sync4j Project, *Sync4j SyncServer Administration Guide*, 04/05/05, http://download.forge.objectweb.org/sync4j/sync4j_syncserver_admin_guide-2.2.pdf

[12] The Sync4j Project, *Sync4j SyncConnector Exchange 1.2 Admin Guide,* http://download.forge.objectweb.org/sync4j/sync4j_syncconnectorexchange_admin_gui de-2.2.pdf

[13] Wireless Application Protocol Forum Ltd., *Wireless Markup Language (WML) Document Type Definition*, version 1.1, 1998-1999, http://www.wapforum.org/DTD/wml_1_1.dtd

[14] J.A. Audestad, *Kompendium TTM 4125 Distribuert prosessering og mobilitet*, Trondheim: Tapir Akademisk Forlag Kompendieforlaget, 2004

Appendix A Implementation

A.1 Servlet

The servlet, PDIServlet, was included in the Sync4j bundle and is extended with search functionality and possibility to save a vCard/iCalendar representation of the files. A number of variables relating to the information stored in the database has also been added (e.g. home and work address, birthday, job title, department, role etc.). All the original files are included on the enclosed CD, in the folder *src\pimweb-old*. JavaDoc is also generated, located in *src\pimweb\javadoc*.

Methods edited by the author:

- processRequest()
- getContact()
- updateContact()

Methods added by the author:

- viewSearchResults()
- cleanTelNumber()
- sortCalendars()
- compare() (In nested class CalendarComparator)
- getDisplayDateFromUTC()
- getUTCDate()

Search functionality

The search function is implemented in such a way that the user can enter several words in a text field, and all of these words must be substrings of the corresponding fields in the database. Figure A.1 shows an excerpt of the method viewSearchResults().

```
…
for(int i=0; i<size; ++i) {
  HashMap item = getCalendar(files[i]);
  StringTokenizer querysumm         = new StringTokenizer(
   (String)request.getParameter(query + PARAM_SUMM           ).toLowerCase());
  StringTokenizer querylocation     = new StringTokenizer(
   (String)request.getParameter(query + PARAM_LOCATION       ).toLowerCase());
…
  boolean match = true;
  while (querylocation.hasMoreTokens()) {
    if (location.toLowerCase().indexOf(querylocation.nextToken()) == -1) {
      match = false;
      break;
    }
    else match = true;
  }
  if (match) {
    while (querysumm.hasMoreTokens()) {
      if (summ.toLowerCase().indexOf(querysumm.nextToken()) == -1) { match = false; break; }
      else match = true;
    }
  }
…
}
```

Figure A.1: viewSearchResults() method

Each file in the respective folder (contacts/calendar) is being checked for its contents (for loop). In the method excerpt in Figure A.1, the entered strings in the summary and location fields are tokenized, and each token is compared to the corresponding content of each of the files in the database. If one of the tokens is not a substring, `match` is set to false and the next item in the for loop is checked. If `match` remains true at the end of the for loop, the current item will be added to the search results.

JSP pages

The JSP pages included in the demo have also been modified. Table A.1 displays the pages in use.

Table A.1: JSP files

Page name	Function
add.jsp	Add a new contact
addca.jsp	Add a new event
getical.jsp	Get the iCalendar representation of an event
getvcard.jsp	Get the vCard representation of a contact
main.jsp	Main page, loads the other pages
menu.jsp	The front menu
results.jsp	View contact search results
resultsca.jsp	View event search results
search.jsp	Contact search page
searchca.jsp	Event search page
view.jsp	View contact details
viewca.jsp	View event details

menu.jsp is the front menu, and all other JSP files are displayed through main.jsp, see Figure A.2. To make the user able to download an iCalendar or vCard, the content types are set accordingly.

```
<% String main = request.getParameter("main");
String task = request.getParameter("task");
if (task != null && task.equalsIgnoreCase("vcard")) {
  response.setContentType("text/x-vcard");
}
else if (task != null && task.equalsIgnoreCase("ical")) {
  response.setContentType("text/calendar");
}
%><jsp:include page="<%= main %>"/>
```

Figure A.2: main.jsp for web interface

The method processRequest() is responsible for processing the HTTP GET and POST variables, and makes sure that the correct JSP page is displayed. Figure A.3 shows an excerpt that makes sure that the user is pointed to the getvcard.jsp file when the GET/POST variable task equals "vcard".

```
…
public final static String TASK_VCARD    = "VCARD";
…
public final static String VIEW_VCARD    = "/getvcard.jsp";
…
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException {
…
  else if (task.equalsIgnoreCase(TASK_VCARD)) {
    viewContact(request);
    next = VIEW_VCARD;
  }
…
  moveToNextView(request, response, next);
…
}
```

Figure A.3: processRequest() method

A.2 WML

For the mobile phone interfaces, the JSP files have been rewritten in WML format. main.jsp is also changed to set the content type to "text/vnd.wap.wml". All files related to the mobile phone interface are located on the enclosed CD, in the folder *src\wap*.

The only change to the PDIServlet code is in the updateContact() method. If a user modifies a contact using his/her mobile phone, the fields that are not included in the WML page (salutation, middle name, suffix, nickname, note and birthday) would be reset, i.e. blank. The processing of these fields was therefore deleted from the updateContact() method for the mobile phone interface.

The example WML card in Figure A.4 is included in the file search.jsp.

```
<wml>

<card id="card1" title="Telenor MRS">
<p align="center">SEARCH FOR CONTACTS<br /><br/></p>
<p>
<b><fmt:message key="msg.fullname"    /></b><br /><input type="text" name="query_fullname"
value=""/><br/>
<b><fmt:message key="msg.org"         /></b><br /><input type="text" name="query_org"
value=""/><br/>
<b><fmt:message key="msg.dep"         /></b><br /><input type="text" name="query_dep"
value=""/><br/>
```

```
<b><fmt:message key="msg.title"       /></b><br /><input type="text" name="query_title"
value=""/><br/>
<b><fmt:message key="msg.role"        /></b><br /><input type="text" name="query_role"
value=""/><br/>
<b><fmt:message key="msg.tel"         /></b><br /><input type="text" name="query_tel"
value=""/><br/>
<b><fmt:message key="msg.address"     /></b><br /><input type="text" name="query_fulladdress"
value=""/><br/>

<br/>

<anchor title="Search">Search
<go href="main.jsp?main=main&demo=contact&task=results" method="post">
<postfield name="query_fullname"    value="$(query_fullname)"                       />
<postfield name="query_org"         value="$(query_org)"                            />
<postfield name="query_dep"         value="$(query_dep)"                            />
<postfield name="query_title"       value="$(query_title)"                          />
<postfield name="query_role"        value="$(query_role)"                           />
<postfield name="query_tel"         value="$(query_tel)"                            />
<postfield name="query_fulladdress" value="$(query_fulladdress)"                    />
<postfield name="task"              value="$(task)"                                 />
<postfield name="main"              value="main"                                    />
<postfield name="item"              value="<%= System.currentTimeMillis() %>"       />
<postfield name="db"                value="<c:out value="${param.db}"/>"            />
</go></anchor>

</p>
<do label="Tilbake" type="prev"><prev/></do>
</card>

</wml>
```

Figure A.4: WML card included in search.jsp (Contact search)

A.3 Screenshots

This chapter includes screenshots of various web user interfaces. Screenshots for contacts and events are included. Figure A.5 shows the front menu.

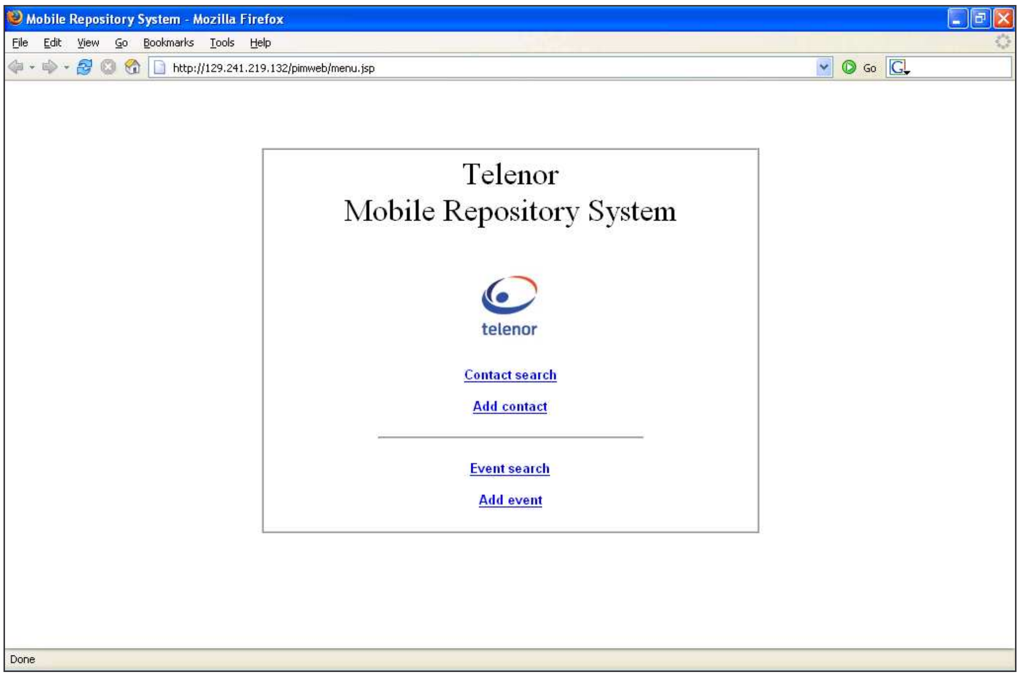
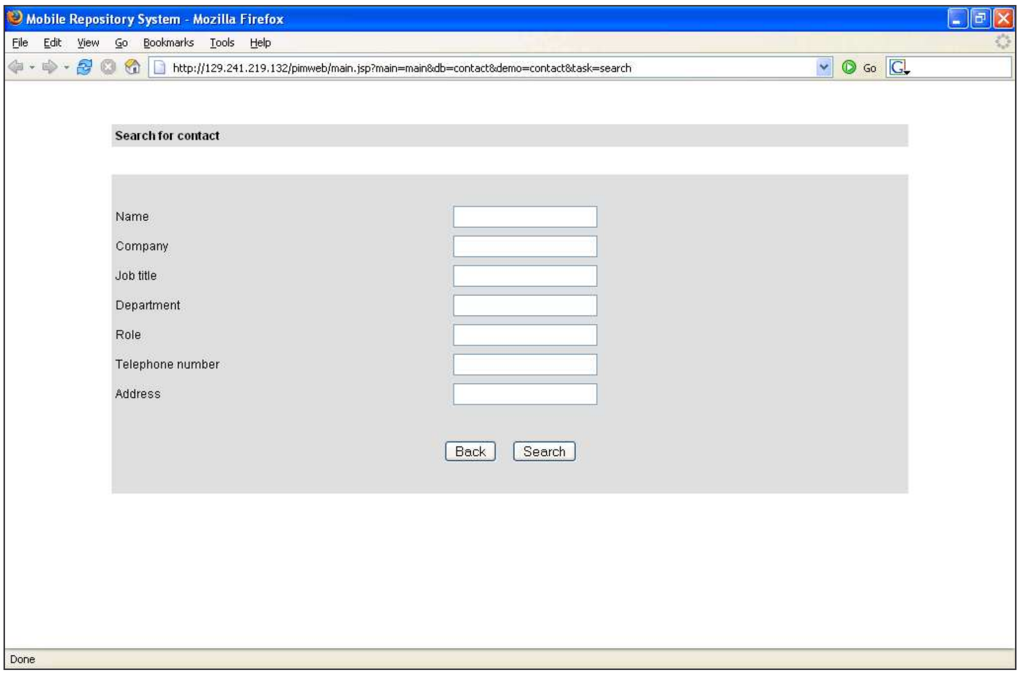

Mobile Repository System - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://129.241.219.132/pimweb/menu.jsp
Go
Telenor
Mobile Repository System
telenor
Contact search
Add contact
Event search
Add event
Done

Figure A.5: Front menu

A.3.1 Contacts

The following screenshots are included:

- Figure A.5 Front menu
- Figure A.6 Search for contact
- Figure A.7 Add contact
- Figure A.8 Contact search results
- Figure A.9 Contact details


Mobile Repository System - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
Go
Done

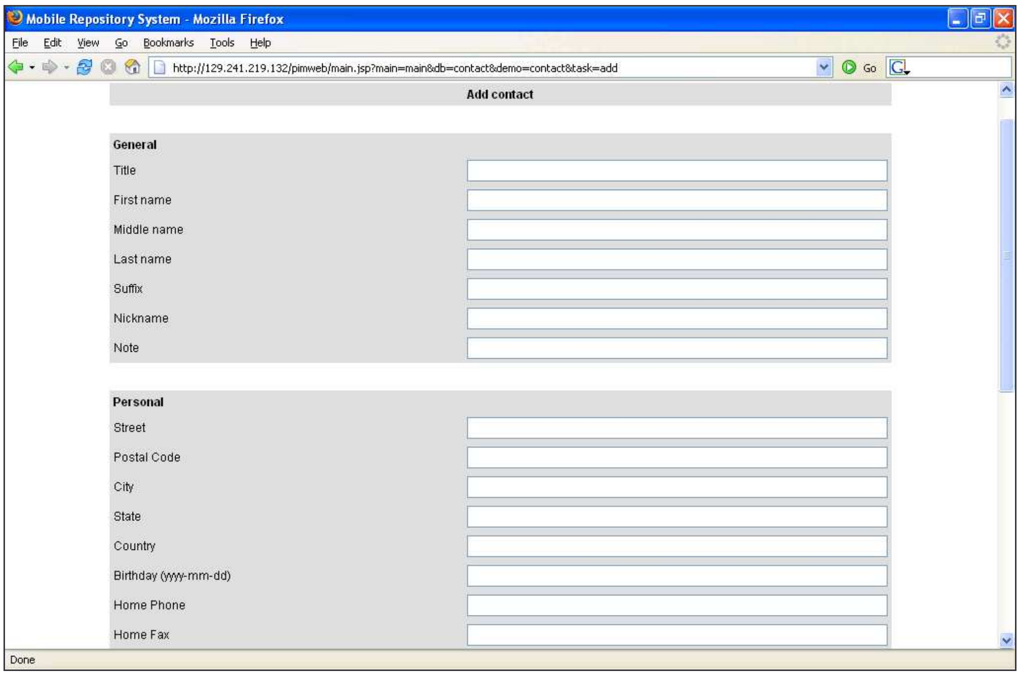
http://129.241.219.132/pimweb/main.jsp?main=main&db=contact&demo=contact&task=search

Search for contact

Name	
Company	
Job title	
Department	
Role	
Telephone number	
Address	

Back Search

Figure A.6: Search for contact

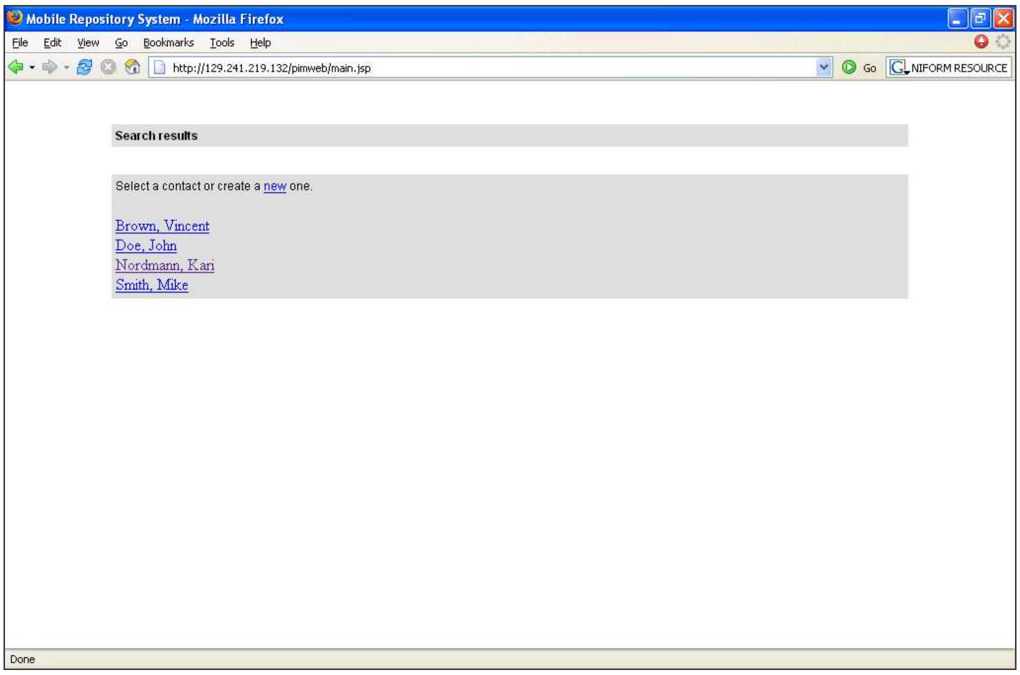

Mobile Repository System - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
Go
Done

http://129.241.219.132/pimweb/main.jsp?main=main&db=contact&demo=contact&task=add

Add contact

General	
Title	
First name	
Middle name	
Last name	
Suffix	
Nickname	
Note	
Personal	
Street	
Postal Code	
City	
State	
Country	
Birthday (yyyy-mm-dd)	
Home Phone	
Home Fax	

Figure A.7: Add contact

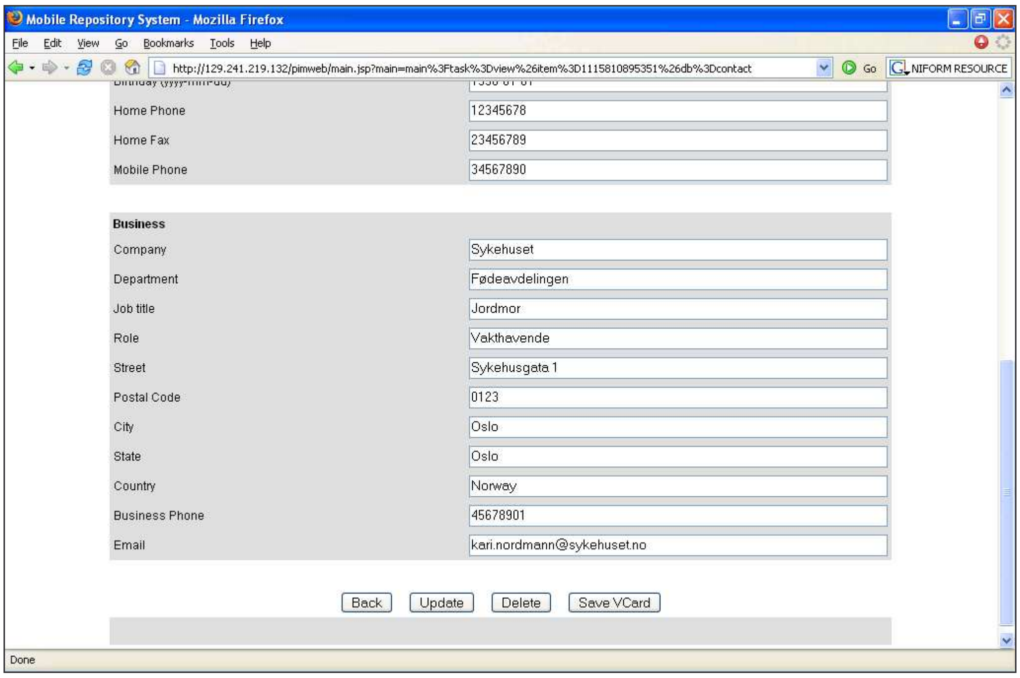

Mobile Repository System - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://129.241.219.132/pimweb/main.jsp
Go
NIFORM RESOURCE
Done

Search results

Select a contact or create a new one.

Brown, Vincent
Doe, John
Nordmann, Kari
Smith, Mike

Figure A.8: Contact search results


Mobile Repository System - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://129.241.219.132/pimweb/main.jsp?main=main%3Ftask%3Dview%26item%3D1115810895351%26db%3Dcontact
Go
NIFORM RESOURCE
Back
Update
Delete
Save VCard
Done

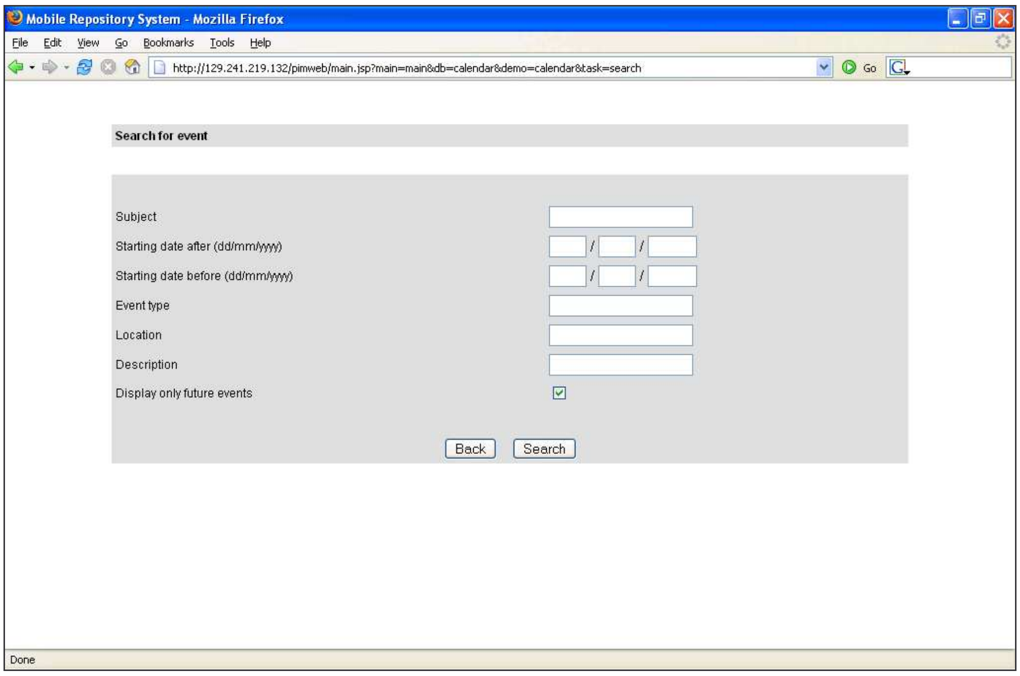
Home Phone	12345678
Home Fax	23456789
Mobile Phone	34567890
Business	
Company	Sykehuset
Department	Fødeavdelingen
Job title	Jordmor
Role	Vakthavende
Street	Sykehusgata 1
Postal Code	0123
City	Oslo
State	Oslo
Country	Norway
Business Phone	45678901
Email	kari.nordmann@sykehuset.no

Figure A.9: Contact details

A.3.2 Events

The following screenshots are included:

- Figure A.10 Search for event
- Figure A.11 Event search results
- Figure A.12 Add event
- Figure A.13 Event details

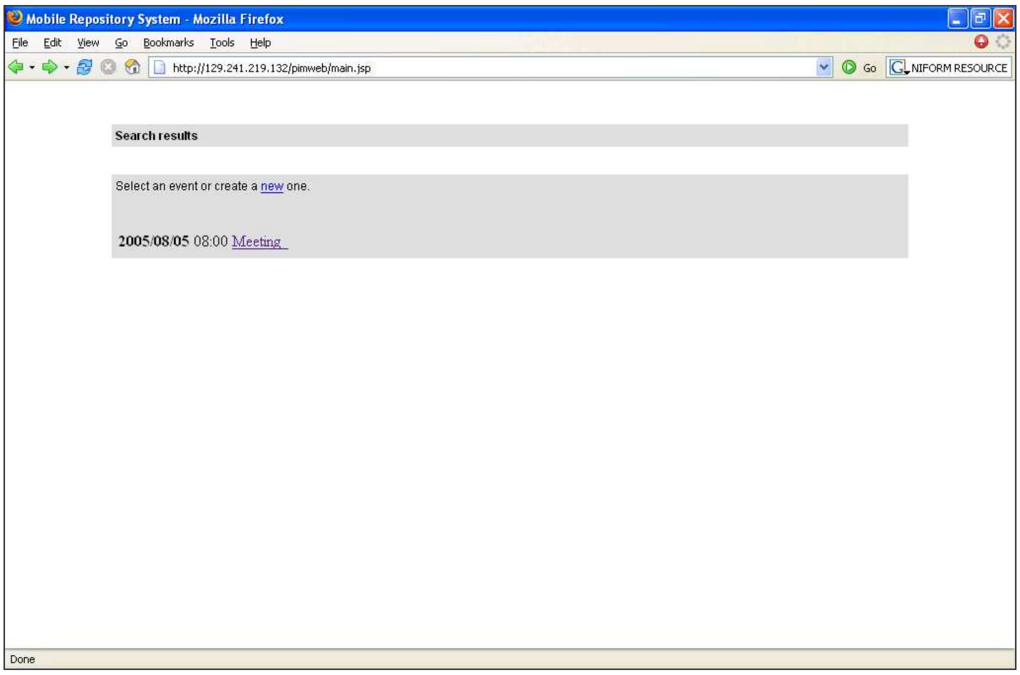

Mobile Repository System - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://129.241.219.132/pimweb/main.jsp?main=main&db=calendar&demo=calendar&task=search
Go
Done

Search for event

Subject	
Starting date after (dd/mm/yyyy)	/ /
Starting date before (dd/mm/yyyy)	/ /
Event type	
Location	
Description	
Display only future events	✓

Back Search

Figure A.10: Search for event

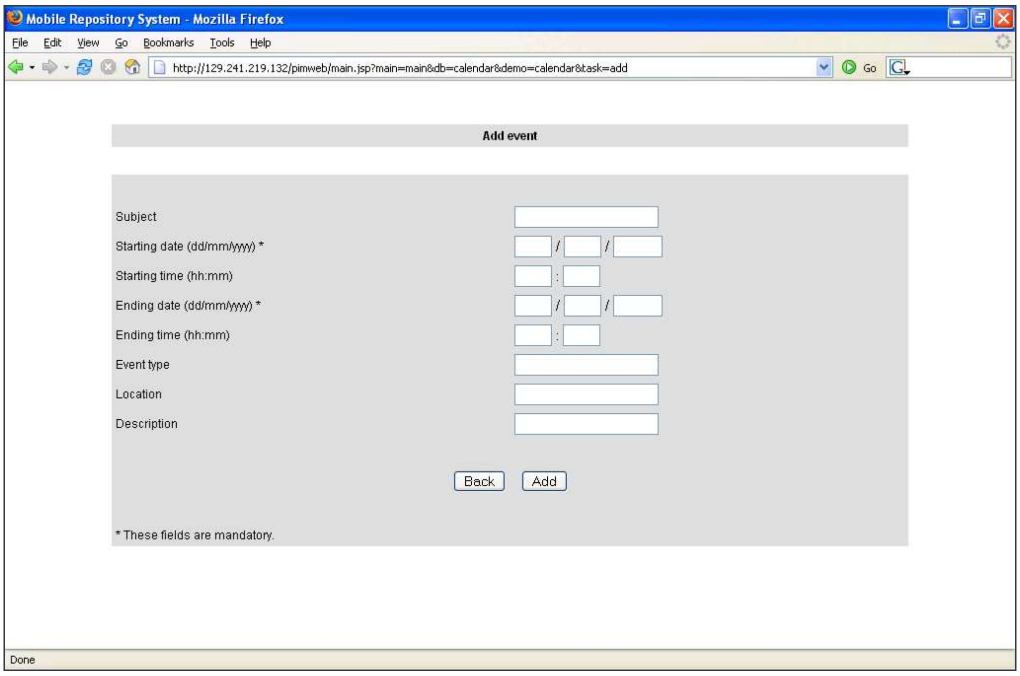

Mobile Repository System - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://129.241.219.132/pimweb/main.jsp
Go
NIFORM RESOURCE
Done

Search results

Select an event or create a new one.

2005/08/05 08:00 Meeting

Figure A.11: Event search results


Mobile Repository System - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://129.241.219.132/pimweb/main.jsp?main=main&db=calendar&demo=calendar&task=add
Go
Done

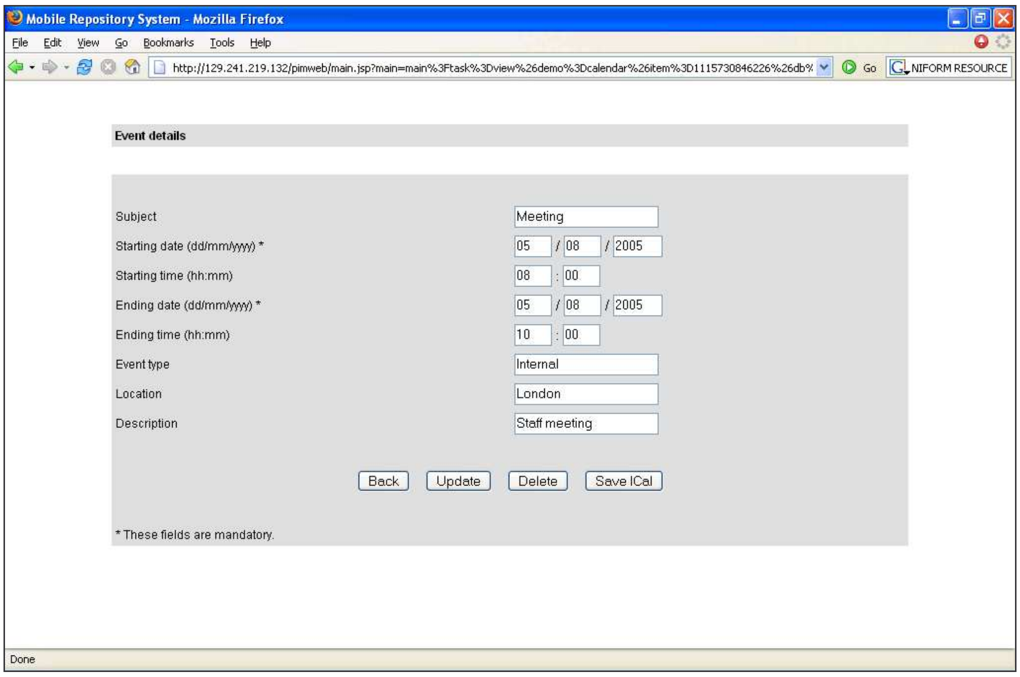
Add event

Subject
Starting date (dd/mm/yyyy) *
Starting time (hh:mm)
Ending date (dd/mm/yyyy) *
Ending time (hh:mm)
Event type
Location
Description

Back Add

* These fields are mandatory.

Figure A.12: Add event


Mobile Repository System - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://129.241.219.132/pimweb/main.jsp?main=main%3Ftask%3Dview%26demo%3Dcalendar%26item%3D1115730846226%26db%
Go
NIFORM RESOURCE
Back
Update
Delete
Save ICal
Done

Event details

Subject	Meeting
Starting date (dd/mm/yyyy) *	05 / 08 / 2005
Starting time (hh:mm)	08 : 00
Ending date (dd/mm/yyyy) *	05 / 08 / 2005
Ending time (hh:mm)	10 : 00
Event type	Internal
Location	London
Description	Staff meeting

* These fields are mandatory.

Figure A.13: Event details

Appendix B User Manual

All the installation steps listed in Appendix B must be completed before the devices can access the SyncServer. At the time of writing, the system is up and running using the IP address specified in the following steps.

B.1 PDA

The following steps are applicable for PDAs with Windows Mobile operating system.

1. Download and install SyncClient PIM Pocket PC /Windows Mobile 1.6.6 on the PDA. The installation file is available on http://sync4j.funambol.com/main.jsp?main=download_stable.
   The file is also available on the enclosed CD (install/syncclient-pim-ppc-1.6.6.exe)
2. Run the program, and choose Edit >Communication Settings.

SyncServer URL:	*http://129.241.219.132/sync4j/sync*
Username:	*employee*
Password:	*employee*

3. Choose Edit > Synchronization Settings.
   Check *Contacts* and/or *Calendar*.
4. Choose Remote Settings.

Contacts:	*./exchange-contacts-pda*
Calendar:	*./exchange-calendar-pda*

   For synchronizing with the file system database (central repository) instead of with the Exchange server, use instead ./sifcontact and ./sifcalendar, respectively.

B.2 Mobile phone

For accessing the central repository, add a bookmark with the address http://129.241.219.132/wap/index.wml.

The following synchronization configuration steps are applicable for Sony Ericsson k700i (with Norwegian software).

1. (optional) Go to "Telefonliste" (menu item row 3, column 2), choose "Valg > Avansert > Synkr.rekkefølge". Check "På etternavn".
2. Go to "Kommunikasjon" (menu item row 4, column 1), choose "Synkronisering > Ny konto".
3. *Navn:* *<anything>*

   Serveradresse: *http://129.241.219.132/sync4j/sync*

   Brukernavn for server: *employee*

   Serverpassord: *employee*

   Tilkobling: *<any valid WAP connection>*
4. Choose "Programmer". Check "Telefonliste" and "Avtaler".
5. Choose "Programinnstilling > Telefonliste".

   Databasenavn: *./exchange-contacts*

   Brukernavn: *<not applicable>*

   Passord: *<not applicable>*

   For synchronizing with the file system database (central repository) instead of with the Exchange server, use instead ./contact
6. Choose "Programinnstilling > Avtaler".

   Databasenavn: *./exchange-calendar*

   Brukernavn: *<not applicable>*

   Passord: *<not applicable>*

   For synchronizing with the file system database (central repository) instead of with the Exchange server, use instead ./calendar

Appendix C Installation Steps

In this appendix, the steps to install, configure and administrate the Sync4j SyncServer are listed. "C:\Program Files\Sync4j" refers to the installation directory of the current implementation, and must be changed accordingly. In the same manner, IP address 129.241.219.132 is the address of the server, and must also be changed.

Install SyncServer:

1. Download and install Sync4j bundle 2.2 for Windows, available from http://sync4j.funambol.com/main.jsp?main=download_stable.
   The file is also available on the enclosed CD (install/sync4j-2.2b4.exe)

Reconfigure the server to use port 80:

2. In `C:\Program Files\Sync4j\tools\tomcat\conf\server.xml`, change the connector port to 80:
   ```
   <!-- Define a non-SSL Coyote HTTP/1.1 Connector on port 8080-->
   <Connector port="80"
   ```
3. In `C:\Program Files\Sync4j\syncserver-4.0.8\config\Sync4j.properties`, change the server address:
   ```
   server.uri=http://129.241.219.132/sync4j/sync
   ```
4. In `C:\Program Files\Sync4j\syncserver-4.0.8\install.properties`, change the server address:
   ```
   server-name=http://129.241.219.132/sync4j/sync
   ```

Add Exchange module:

5. Download and unzip SyncConnector Exchange 1.2.4, available from http://sync4j.funambol.com/main.jsp?main=download_stable
   The file is also available on the enclosed CD (install/syncconnector-exchange-1.2.4.zip)
6. Copy syncconnector-exchange-1.2.s4j to syncserver-4.0.8\modules
7. In `C:\Program Files\Sync4j\syncserver-4.0.8\install.properties`, add the Exchange modules to the list of modules to install:
   ```
   modules-to-install=foundation-1.5.1,pdi-1.3,pimweb-1.1.1,
   syncconnector-exchange-1.2
   ```

Reinstall SyncServer:

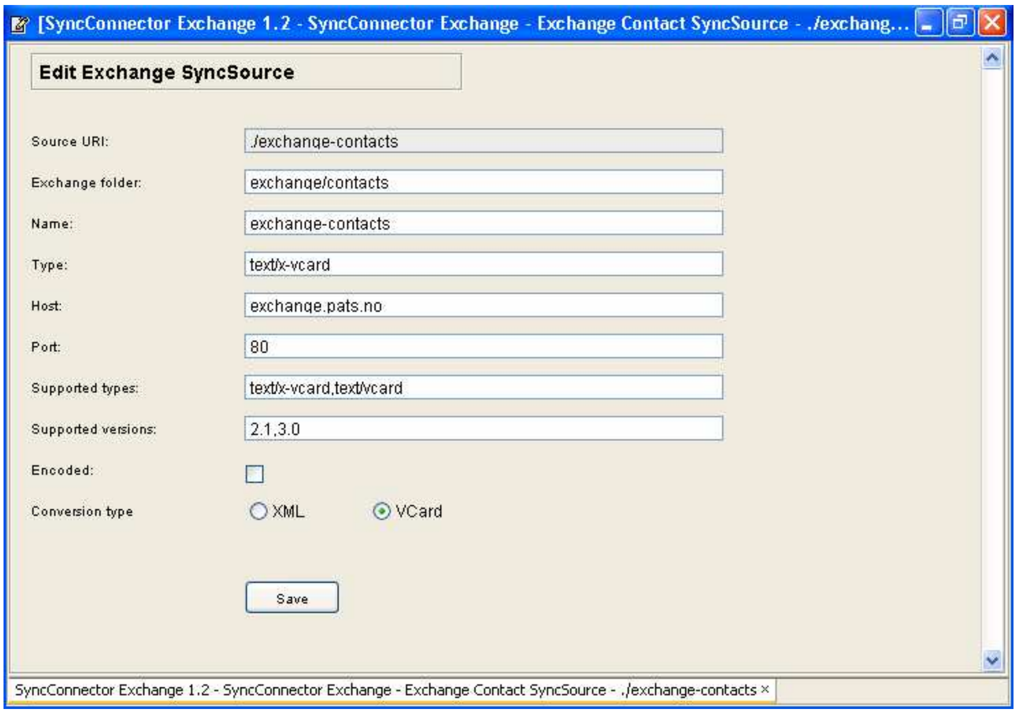
8. Start the Sync4j server (Start > Programs > Sync4j > SyncServer > Start)
9. Open a command shell, and set the relevant environment variables:
   `SET J2EE_HOME=C:\Program Files\Sync4j\tools\tomcat`
   `SET JAVA_HOME=C:\j2sdk1.4.2_05` (or similar)
10. Change the command line path to `C:\Program Files\Sync4j\syncserver-4.0.8`
11. Type `bin\install tomcat50` to re-install the SyncServer.
12. Re-start the SyncServer. Due to a system tray bug when changing the port from the default value 8080, this must be done from the command shell:
    `C:\Program Files\Sync4j\tools\bin\stopall.cmd`
    `C:\Program Files\Sync4j\tools\bin\startall.cmd`
    Note: The system tray icon will remain red after it has started. Verify that the server has been properly started by accessing http://localhost/sync4j.

User interfaces:

13. Replace the folders `C:\Program Files\sync4j\tools\tomcat\webapps\pimweb` and `C:\Program Files\sync4j\tools\tomcat\webapps\wap` with the contents of the folders *pimweb* and *wap*, respectively, located on the enclosed CD
14. Repeat step 12.

SyncAdmin:

15. In the login window, change the port number to port 80.
16. Add Exchange SyncSources (see Figure C.1, Figure C.2 and corresponding note).
17. (Optional) Change log level to ALL.
18. (Recommended) Change password for the Admin account
19. Add device(s) for PDA(s). The device ID can be found when starting the SyncClient PIM Pocket PC (Appendix B.1).
20. Add user *employee* with password *employee*, and principal <employee,sc-pim-ppc>. The username *employee* must also exist in the Exchange server.
21. Add device for mobile phone(s). The device ID is the mobile phone's IMEI number. This can usually be found by pressing *#06# on the mobile phone, and only the first 15 digits should be entered.
22. Add principal <employee,IMEI number>


[SyncConnector Exchange 1.2 - SyncConnector Exchange - Exchange Contact SyncSource - ./exchang...

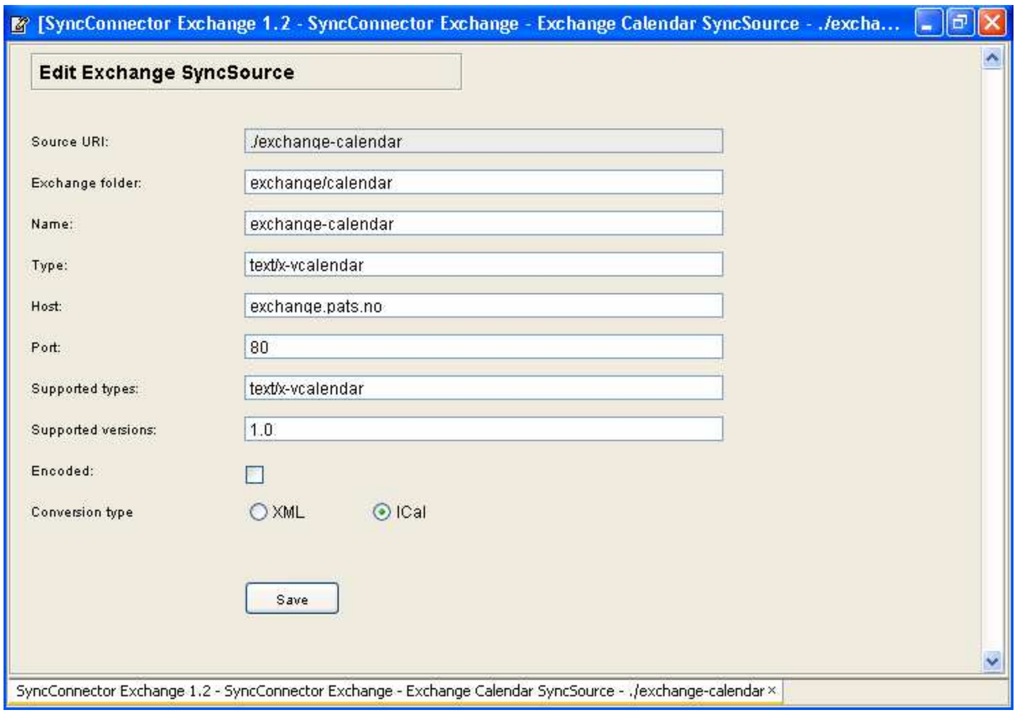
Edit Exchange SyncSource

Source URI:	./exchange-contacts
Exchange folder:	exchange/contacts
Name:	exchange-contacts
Type:	text/x-vcard
Host:	exchange.pats.no
Port:	80
Supported types:	text/x-vcard,text/vcard
Supported versions:	2.1,3.0
Encoded:	☐
Conversion type	XML [✓ VCard]

Save

SyncConnector Exchange 1.2 - SyncConnector Exchange - Exchange Contact SyncSource - ./exchange-contacts

Figure C.1: Exchange Contact SyncSource


[SyncConnector Exchange 1.2 - SyncConnector Exchange - Exchange Calendar SyncSource - ./excha...

Edit Exchange SyncSource

Source URI:	./exchange-calendar
Exchange folder:	exchange/calendar
Name:	exchange-calendar
Type:	text/x-vcalendar
Host:	exchange.pats.no
Port:	80
Supported types:	text/x-vcalendar
Supported versions:	1.0
Encoded:	☐
Conversion type	XML [✓ ICal]

Save

SyncConnector Exchange 1.2 - SyncConnector Exchange - Exchange Calendar SyncSource - ./exchange-calendar

Figure C.2: Exchange Calendar SyncSource

Note: The configurations in Figure C.1 and Figure C.2 are valid for use with mobile phones. For synchronization with a PDA, using the SyncClient PIM Pocket PC / Windows Mobile client, **the conversion type needs to be set to XML and encoded must be checked**. The *Exchange folder* value must remain the same, but the *Name* value must be changed, since the names must be uniquely identifiable. In the current configuration, two additional SyncSources have been added, ./exchange-contacts-pda and ./exchange-calendar-pda.

Appendix D Enclosed CD

The following directories can be found on the enclosed CD:

install	Sync4j SyncServer bundle 2.2 beta 4
	SyncClient PIM Pocket PC / Windows Mobile 1.6.6
	SyncConnector Exchange 1.2.4
log	SyncServer log file example
references	The references used that are electronically available
report	This report in PDF format
src	JSP pages and servlet code for web and WAP (located in *pimweb* and *wap*, respectively), as well as the original files bundled with Sync4j (located in *pimweb-old)*. The servlet code is located in *WEB-INF/classes//PDIServlet.java*.