

Synchronization Concept for phpGroupware

Markus Kämmerer, mkaemmerer@probusiness.de

Matthias Leonhardt, mleonhardt@probusiness.de

<http://www.probusiness.de>
<http://www.probusiness.de/projekte/phpgroupware/>

Last edited: 2004-06-20

Table of contents

1. General.....	3
1.1. Introduction.....	3
1.2. Seven layer concept.....	3
1.3. phpGW Module layer.....	3
1.1.1. Implementation details.....	4
1.1.2 ACL.....	4
1.4. phpGW API.....	5
1.2.1. vCard/vCalendar Interface.....	5
1.5. phpGW SyncModule.....	5
1.6. SyncModule IPC.....	6
1.7. SyncML Server.....	7
1.5.1. Transport Layer between SyncML Server and Client.....	7
1.8. SyncML Client.....	7
1.6.1 SyncML Client for end user devices without SyncML support.....	7
1.6.1.1 Palm devices on Windows.....	7
1.6.1.2 Palm devices with Linux.....	8
1.6.1.3 PocketPC devices.....	8
1.6.1.4 Offline version of phpGW.....	8
1.6.1.5 Unknown devices	9
1.9. End User Device.....	9
2.Layer independent considerations.....	10
2.1.Record conflict handling.....	10
2.2.SyncML sync types.....	10
2.3.Device Management.....	11
2.4.Record Mapping.....	11
3.Databases.....	12
3.1. SyncDevices.....	12
3.2 Mapping.....	12
3.3. Sources.....	13
3.5. Conflicts.....	13
3.6. SyncChannels.....	13
4.Link list and tables.....	15
4.1 SyncML, http://www.syncml.org	15
4.2. Coldsync, http://www.coldsync.org	15
4.3. Kolab Project, http://kolab.kde.org/	15
4.4. Outlook Connector Project, http://otlkcon.sourceforge.net/	15
4.5. Links in Wiki from phpGroupware, http://docs.phpgroupware.org	16
4.6. Sync4j, http://sync4j.funambol.com/main.jsp?main=theproject	16
4.7. Misc Links.....	16
4.8. SyncML capable devices.....	18
4.9. Misc SyncML: http://www.nexthaus.com	18
5.Installation.....	18
6.Glossary.....	19
1.Milestones.....	20
1.History.....	21

1. General

1.1. Introduction

// to be written

1.2. Seven layer concept

This concept uses a seven layer implementation (see “Sync Concept Module Diagram”):

Layer 1: phpGW Module Layer

Layer 2: phpGW API

Layer 3: phpGW SyncModule

Layer 4: SyncModule IPC

Layer 5: SyncML Server

Layer 6: SyncML Client

Layer 7: End User Device

Between each layer lies a well defined interface. Every layer may be replaced by another piece of software that implements the same in- and output interface like our original layer (except layer 1-2).

The following chapters will describe each layer:

1.3. phpGW Module layer

Each phpGW API Module may or may not have some sort of data to synchronize. With the exception of addresses and contacts there is no common standard how this data are represented in SyncML. It is the task of every own module to collect all data to synchronize. Every Module has its own distinguish name (= server name), which is needed to connect the phpGW Module to the end user module.

SyncML specifies a standard format for some common modules:

<i>phpGW module</i>	<i>server name</i>	<i>MIME-Type</i>	<i>SyncML specification</i>
Address Book	Contact	text/vcard	must be able to produce vCard 2.1 and optionally vCard 3.0 data
Calendar	Calendar	text/vcalendar	must be able to produce vCalendar 1.0 and optionally iCalendar 2.0.
Memos	Memo	text/plain	must support text/plain
Tasks	Task		must support vTodo 1.0

<i>phpGW module</i>	<i>server name</i>	<i>MIME-Type</i>	<i>SyncML specification</i>
E-Mail	EMail		must support message/rfc822, message/rfc2822 and message/rfc2045

Support for the standard SyncML data representation is included in phpGW for Address Book and Calendar already. Some improvements are necessary, see the document “Class Overview vCard and iCalendar”

1.1.1. Implementation details

In our first draft we planned to add global records to detect, which application module is able to synchronize. In our current implementation this is not necessary anymore. The list of modules, which are able to be synchronized are detected through records in the `phpgw_syncsources` database. Default values for all current supported application modules are added during setup of sync module.

The application module must support time stamp handling for changed records, this allows us to detect changed records after a specific time of synchronization (this time stamps are recorded in sync module for every source and device).

Every application module which want to be able to synchronized must support an IPC interface with a couple of different functions. The IPC interface is generic for all modules and will be called through the IPC manager from the sync module. More documentation you'll find in the IPC documentation (check out module 'ipc_test_suite' on Savannah HEAD branch).

Note: „each record to sync should have also an UID, a unique identifier as the SyncML server has to maintain a table of SUID/CUID (Server UID and Client UID) in order to facilitate the mappings of the records. We have to discuss if the phpGW record_id suffice for this. The UID needs only to be unique inside every phpGroupware module (Layer 1), because the record is always identified by the UID/source name pair.

1.1.2 ACL

The phpGW module is the only layer that knows everything about its data and the user (with the help of the API). Every call to a function in this layer is done while the current sync user is logged into phpGW. It is the responsibility of this layer to check the ACLs for this user and send only the items which are allowed to access by this user. Later, when the device is on air the ACLs have to be checked again, because for every record the ACLs may have changed meanwhile and access to existing records is not allowed anymore. In case an existing record is not allowed to access there are three possible ways to handle this:

- synchronize this record the normal way
- delete the record on the end user device
- don't delete this record but do not sync anymore with new versions from the server

It is the task of the module to decide the correct handling.

1.4. phpGW API

Inside the phpGW API we need support of a generic API. Every module has to inform the API about its possibility to do synchronization and its Server name. In addition the API has support functions not directly related to synchronization like v* helper functions.

Inside the API there is also the IPC manager, which is used to access IPC functions.

1.4.1 vCard/vCalendar Interface

The creation of vCard/vCalendar format normally is the task of the address book and Calendar-Module. But this two formats are so common for sharing addresses and date items that the support of this format is mandatory. In the current phpGW these functions are located in the API, not in the modules.

1.5. phpGW SyncModule

This part of the concept is the largest one. This module has to do the following tasks:

- handle the mapping of GUID and LUID
- store the devices with all parameters for every user (DM = Device Management)
- conflict handling: save records which issue a conflict and wait for confirmation from user, which records has to be saved or merged
- end user interface that shows conflicts and allows merging/selecting the right record
- end user interface for adding/removing devices (can be more than one device per user)

Problems to solve:

- different end user devices handle the changed/dirty flag different. Some devices (like PalmOS devices) use only one flag for each record. Other devices use one flag for each part of the record. This helps for automatic merging of records. But at this stage we will only implement one dirty flag for one record
- some data have to be synchronized only for one device (like private data for one device, business data for other devices)

There are two complete ways to handle dirty flags for fast syncs.

1. The phpGW module which supports sync sends an update request to the SyncModule every time an Item is changed. This is simple, because the only data needed are: SourceID and GUID. The SyncModule has to set the dirty flag in the table Mapping to true where SourceID and GUID are identical with the call parameters. This way, all devices 'know' about the changed record on the server next sync. Every time the devices are synced the dirty flags for this device are cleared. This way, one update call from the module may set 0-n dirty flags depending on how many devices have this record (these are not necessary all records because every device may have different filters which records are to be synchronized). This way is fast, because the dirty flags are ready almost when the devices start syncing (there is only one select to the table which is

needed always). At the end of a successful sync the sync flag is set to false. New records are not handled this way, because there is no mapping entry for them. At sync time the SyncModule still has to ask the phpGW module to send all new records after a specific date (with the help of the SyncTimes table).

2. When the SyncModule receives a synchronization request from the client, one call is made from the sync module to the phpGW module with the time stamp of the last successful sync of this source. The phpGW module returns an in-memory list of in items (one GUID for every) that changed after this time stamp. The SyncModule has to compare all mappings from the Mapping table with the returned values. If a GUID is in both the Mapping table and the item list the record was changed and we need to synchronize this record. We need an extra database to store DeviceID, SourceID and Timestamp (table phpgw_syncchannels). This way is slower than way one for two reasons: 1. The phpGW module has to determinate the changed records when the sync device is 'on the air'. 2. The in memory list of items may be large and has to be compared with PHP with the result of the select result of Mapping. With way this mapping is done entirely through the database.

Slow sync: all records from either the server and the client are transfered to each other. This is done by deleting all GUID-LUID mapping before the modification detection.

In the current implementation part II is used. No dirty flag mapping is done, only the timestamps are saved.

1.5.1 GUI for Layer 4

Tasks:

- conflict management
- list all devices which are connected by the actual user, allow to change UserID and Password for each device

1.6. SyncModule IPC

This layer is small compared with the other layers. This layer handles the communication between the phpGW SyncModule and the SyncML Server. At the first stage of implementation the SyncML Server Component (Part 5) is Sync4j which is written in Java and uses a JBoss Server. In later implementations the Layer may interface with a self written PHP SyncML Server.

To allow flexible communication with various SyncML servers this layer is implemented with the phpGW IPC interface. This interface allows to call the functions of this layer either through XML-RPC (in our case from Sync4j) or inside from phpGW from any other module (e.g. an internal SyncML Server written in PHP).

The sync module IPC is located in /sync/inc/class.ipc_sync.inc.php, the main IPC classes are in phpgwapi/inc/class.ipc_.inc.php and class.ipc_manager.inc.php. Maintainer of the IPC classes is Dirk Schaller <Dschaller@probusiness.de>.

1.7. SyncML Server

The SyncML Server is used for communication with the SyncML client through one of the supported transport bindings (see 1.5.1). Layer 5 is the last layer on the Server. First

implementations will use Sync4j SyncML Server. Later implementations may use a native SyncML server written in PHP.

1.5.1. Transport Layer between SyncML Server and Client

The Transport Layer between the SyncML Server and the SyncML Client is entirely specified in the SyncML Protocol. In Version 1.1 of SyncML the following three transport bindings are included:

- HTTP: This is the easiest way to implement the transport binding. Sync4j only supports HTTP at this time. Most available end user devices support HTTP. There is a reference implementation written in PHP. A secure version is possible with HTTPS
- WSP uses binary compressed XML over WAP
- OBEX is used with Bluetooth

SyncML is not restricted to these protocols. Own implementations may use other transport bindings like WeBDAV or even POP3. Most SyncML clients support only HTTP, so we should at least support HTTP as transport binding.

1.8. SyncML Client

The SyncML Client communicates through the transport binding with the SyncML Server (Layer 5) and is located near or inside the end user device. Some newer devices have builtin support for SyncML (mainly Sony Ericsson P800, some Nokia Smartphones and other SymbianOS devices). With such devices no more special software is needed for synchronization with phpGW (at least with Contacts and Calendar). Even jengo's xmlrpc enabled toaster may work, at least according to Dave Hall's mail :-).

1.6.1 SyncML Client for end user devices without SyncML support

Palm Devices use their own synchronization system. For Windows there is the official synchronization software 'Hotsync' available. There is no official Linux support from PalmSource.

1.6.1.1 Palm devices on Windows

On Windows the software 'Hotsync' is used for synchronization and backup of any Palm devices. It is possible to use more than one device with one Hotsync. Hotsync is possible through a serial line (cable or IR), USB, Bluetooth or Network (TCP/IP). It is possible to add own conduits to extend existing synchronization (like Contacts and Calendar) or add own modules (e.g. for additional phpGW modules with self written counterpart on Palm).

There are more than one high quality Delphi libraries available to add sync support for Palm (libs for parsing vCalendar, libs to write a HotSync Conduit, libs for SyncML). We need to implement an

minimal SyncML Client within Delphi to allow Hotsync with Palm.

Because this part of software is entirely located on layer 7 it is possible that a generic (maybe commercial) SyncML Client will be developed by other software developers. There is no direct connection or even knowledge necessary to add sync support for Palm with phpGW (at least for the both standard modules calendar and addresses). Known Clients for this are the Clients from Weblicon (uses Hotsync Conduit) and Synthesis (using native Palm Software)

It is possible to add synchronization support for Palm devices without adding SyncML support with connecting directly to layer 4 instead connecting to layer 6. This solution may be easier than adding SyncML support but has its disadvantages. It lacks of much of the modularity and functionality of a complete SyncML client for Palm implementation. In addition, an SyncML Palm conduit may connect to all SyncML servers, not just phpGW.

1.6.1.2 Palm devices with Linux

There is no official support for Linux from PalmSource. The Linux software ColdSync implements Hotsync for Palm, but does not support SyncML. Some little modifications and additions to the Delphi SyncML Client (see 1.6.1.1.1) may be enough to support Hotsync on Linux. The commercial client from Synthesis runs on the PalmOS device itself and is able to do OS independent SyncML over the air.

1.6.1.3 PocketPC devices

PocketPCs do not support SyncML, like Palm devices. There are commercial SyncML clients for PocketPCs available (Weblicon, Synthesis and others, see our list)

1.6.1.4 Offline version of phpGW

Generally speaking the offline version of phpGW is nothing different from any other Sync Client and is handled internally the same as any other sync capable device. Because every SyncML Server includes all components of a SyncML client it should not be that hard to support phpGW <-> phpGW synchronization. Care must be taken that a offline phpGW consists always only a subset of the normal phpGW functionality. In the offline version there is only a subset of modules and a dependent on the module a subset of data in the local database. Offline phpGW should not allow direct synchronization with other end user devices, because this may cause a lost of data or other trouble things. The disadvantage of a SyncML Client is that mobile devices like Palm or Smartphone have to synchronize via the air (with additional costs) and can not synchronize with the notebook directly. At later stages of development there may be some changes.

Offline phpGW has one big disadvantage: because there is no SyncML client written in PHP there (see milestone 8) every offline installation needs an installed JBoss with Sync4j. To write an SyncML client in php for offline synchronization is easier than writing the same for all synchronization processes, because you'll need only a minimal featured server which has only to be certified with one SyncML server (in our case Sync4j).

1.6.1.5 Unknown devices

Unknown devices only need SyncML Client support and should at least be able to synchronize Contacts and Calendars.

1.9. End User Device

Every end user device with SyncML client capabilities connects directly to the SyncML server. Every other devices needs a middleware described in chapter 1.6.

Attention: Some devices like Palm support network sync. In this case the client conduit may not be installed on the client pc, instead the conduit has to be installed on the network server to which the palm connects.

2. Layer independent considerations

2.1. Record conflict handling

If a record is changed on either the end user device and the database for the same time, there will be a conflict during the next synchronization request. There are for solutions for this:

- Server wins: The record from the server overwrites the record on the device with no user invention. The data on the device will be lost
- Client wins: The record from the client overwrites the record on the server database with no user invention. The data on the server database will be lost.
- Duplicate: Every changed record will be copied to the opposite direction. Thus, the record from the server will be copied to the client and otherwise. These records get new IDs and do not overwrite oder data. To allow distinguish between the original and duplicated records the main display information should be changed. This is only possible through level 1 interface.
- Do Nothing: Neither the server or device records are touched. The record from the device will be copied to an internal database on the server. The resolve of the conflict will be done later with user invention.

From Sync4j developers manual:

Once a conflict arises and it is detected, a proper action must be taken. Different policies can be applied:

- User decides: the user is notified of the conflict condition and decides what to do; this strategy, like the following “Client wins” is a bit problematic in a server centric synchronization solution: each user may have the same right to modify an item and one users could not be able to decide whether his/her modification should win over the other ones.
- Client wins: the server silently replaces conflicting items with the ones sent by the client.
- Server wins: the client has to replace conflicting items with the ones from the server.
- Time stamp based: the last modified (in time) item wins
- Last/first in wins: the last/first arrived item wins
- Do not resolve

2.2. SyncML sync types

SyncML specifies more than one sync type:

- Two-way sync
- Slow sync
- One-way sync from client only
- Refresh sync from client only
- One-way sync from server only
- Refresh sync from server only
- Server alert sync

2.3. Device Management

2.4. Record Mapping

The record mapping is done with the table `phpgw_syncmapping` in sync module (layer 3). Every record in phpGW has an unique ID (called GUID). This record has to be unique inside one application module. Because the mapping is always associated with GUID and the source (=application module) the ID may overlapp between different modules. The device itself maintains an LUID for each record. Through this mapping the sync module knows, which records was added, deleted and modified since last synchronization (with the help of the timestamp).

3. Databases

Sync4j uses the database table sync4j_principal which contains the entity that can make a synchronization request. Principal contains a couple of username and DeviceID. In phpGW we use the deviceid as identifier for entity that can make an synchronization. Table DeviceID contains UserID, so it is possible to get all devices for one user. Because all other parameters are device-dependent, not user dependent we use the DeviceID as identifier.

3.1. phpgw_syncdevices

One User may have more than one device. This database handles 1:n Mapping of users to devices. Each device has its URI (with cellular phones this is the EMSI number). In addition it contains the last local and remote anchor.

<i>Fieldname</i>	<i>Description</i>
DeviceID	Auto increment, Primary Key
AccountID	UserID from phpGW from type Integer (phpgw_accounts.account_id), \$GLOBALS['phpgw_info']['user']['account_id'];
URI	ID of the device specified by <Source><LocURL> element of <SyncHdr>, EMSI number on cellular phones, this field is unique
Description	Description of the Device, may be changed by the User in SyncModule
Username/Password	May be empty. Checked against the credentials on the device, will be set at first PnP Synchronization request from unknown device, if Username and Password are valid phpGW login (UserID is set, too)
DevInfo	Cached device information

3.2 phpgw_syncmapping

Contains one record for each item in the remote device. If an item is soft-deleted on a client then the LUID field will be empty.

<i>Field Name</i>	<i>Description</i>
MappingID	auto increment primary key
ChannelID	Combination of DeviceID and SourceID, ID of phpgw_sychannels
LUID	Local (=device owned) record number
GUID	Global (=phpGW) record number

3.3. *phpgw_syncsources*

Every phpGW module has to tell the API that it may synchronize. The field URI contains the name which has to be entered in the end device to specify the sync source (e.g. “Calendar”, Server Name, see chapter 1.1).

<i>Field Name</i>	<i>Description</i>
SourceID	Autoincrement, Primary Key
URI	'ServerName' like “contacts”
ModuleName	name of the phpGW application which handles this source
MimeType	supported (prefered) mime type for this phpgw module
MimeVersion	Version number of supported mime type (e.g. “2.1”)
DisplayName	name of the sync source for display purposes

3.5. *phpgw_syncconflicts*

If the same record is modified both at the device and the server at the same time we have a conflict. If the SyncModule can not resolve this conflict automagically then both records are saved to this table. The user may be able to solve this conflict later. If there is more than one conflict with the same LUID/GUID the time stamp. // !!

<i>Field Name</i>	<i>Description</i>
ChannelID	Points to the DeviceID+SourceID combination
LUID	Original LUID, same as in table SyncMapping
LData	Local data (from the device), in binary raw format
Timestamp	Time when the conflict was detected, because the same record may produce more than one conflict (with multiple devices). Conflicts are solved in in time line (oldest conflicts first)

3.6. *phpgw_syncchannels*

<i>Field Name</i>	<i>Description</i>
ChannelID	Autoincrement, Primary Key
DeviceID	FK to table SyncDevices
SourceID	FK to table SyncSources

<i>Field Name</i>	<i>Description</i>
LastAnchor	A string representing a synchronization event. The format of the string will typically be either a sequence number or an ISO 8601-formatted extended representation, basic format date/time stamp
StartSync/StopSync	Unix Timestamp (Int32)
conflict	How to handle conflict: 1: server record wins 2: client record wins 3: duplicate records 4: do nothing

4. Link list and tables

4.1 SyncML, <http://www.syncml.org>

SyncML ist ein Datensynchronisations-Protokoll und basiert auf XML, inkl. Device-Management. Ein Softwaremanagement wird folgen

Spezifikation und Protokollbeschreibung ist auf der Homepage verfügbar (<http://www.openmobilealliance.org/syncml/downloads.html>).



Verschiedene DTDs sind mit Dokumentation verfügbar devinf.dtd: Device-Information, beschreibt ein entsprechendes Geräte SyncML HTTP Binding beschreibt, wie entsprechende SyncML Messages über HTTP übertragen werden können SyncML Sync Protokoll beschreibt den eigentlichen Synchronisationsprozeß SyncML Representation Protocols enthält ein Standardformat für einige gängige Daten

4.2. Coldsync, <http://www.coldsync.org>

„ColdSync is a tool for synchronizing PalmOS devices (PalmPilot, Palm V, Qualcomm PDQ, etc.) with Unix workstations.“

There is more than one conduits available:

- phpgroupware palm sync (<http://sourceforge.net/projects/phpgwpalmsync/>), A set of coldsync conduits syncing phpgroupware and palm pilots., Alpha-Status, 18kb Perl Script
- LDAPFetch (<http://muspellsheim.net/software/ldapfetch/>) is a ColdSync (www.ooblick.com/software/coldsync) conduit that will update a PalmOS address book with information from an LDAP Server., 3kb Perl Script
- palm-perl: Perl module for reading and writing Palm databases (both PDB and PRC)

4.3. Kolab Project, <http://kolab.kde.org/>

- <http://kolab.kde.org/kolab-plugins.html>
- <http://kontakt.kde.org/>
Work is underway on a framework called Kontact (<http://kontakt.kde.org>) that will unify KMail, KOrganizer, KAddressbook and KNotes into one KParts based application. This will become the standard for KDE based mail and form the next generation KDE Kolab Client.

4.4. Outlook Connector Project, <http://otlkcon.sourceforge.net/>

„The Outlook Connector Project’s aim is to develop extensions for the Microsoft Outlook® email/grouper client that would allow it to use its full functionality with email and calendar servers

other than Microsoft Exchange® email/groupware server.“

Pre-Alpha-Stadium, LGPL

4.5. Links in Wiki from phpGroupware, <http://docs.phpgroupware.org>

- How SyncML works
<http://docs.phpgroupware.org/SyncML>
- SyncInterface, Important! <http://docs.phpgroupware.org/SyncInterface>
- PeerServers, Ideen, wie mehrere phpGWs miteinander verbunden werden können
<http://docs.phpgroupware.org/PeerServers>
- PalmSync:coldsync+perl+xmlrpc
<http://docs.phpgroupware.org/wiki/index.php?domain=default&page=PalmSync>

4.6. Sync4j, <http://sync4j.funambol.com/main.jsp?main=theproject>

The Sync4j project consists of:

a Java class library that implements the SyncML data synchronization protocol a Java-based application framework for building SyncML server applications a standalone SyncML server

- License: identically to JDOM
- Important packages: sync4j.core, sync4j.client, sync4j.server, sync4j.http/wsp/obex, sync4j.tests
- Nokia: Forum with discussion about SyncML
<http://discussion.forum.nokia.com/forum/forumdisplay.php?forumid=31>
- Sync4j Javadoc
<http://sync4j.funambol.com/project/development/javadoc/>

4.7. Misc Links

- The commercial 1.7 software development kit contains the tools required to create commercial Qtopia applications for the Sharp Zaurus. It is intended for use by third-party software developers.
<http://www.trolltech.com/download/qtopia/index.html>
- Making SOAP with Soup http://developer.ximian.com/articles/whitepapers/soap_soup/
- PalmOS Developer Tools
<http://www.palmos.com/dev/tools/>
- Delphi Palm Conduit Library <http://delphi-conduits.sourceforge.net/>
- Powerful combination of a state-of-the-art, 32-bit Pascal compiler and an easy-to-use, adaptable integrated development environment that allows you to quickly build sophisticated applications for the Palm OS <http://www.winsoft.sk/pstudio.htm>
- TurboSync is a set of VCL components and Delphi classes that enable Palm conduits to be easily developed in Delphi. <http://www.tabdee.ltd.uk/Downloads/TurboSync.html>

- SyncML ist ein Plattform- und Übertragungsmedium-unabhängiges Synchronisations-Protokolls
http://www.synthesis.ch/servic_d.php
- Synthesis, SyncML Dienstleitungen http://www.synthesis.ch/servic_d.php
- SyncML tools mit PHP <http://nicolas.bougues.net/syncml/>
- vCard and vCalendar specification <http://www.imc.org/pdi/>
- Using PHP to Make Basic vCalendar/iCalendar Events
<http://www.phpbuilder.com/columns/chow20021007.php3>
- libical bindings for PHP v0.9 <http://www.a-s-i.com/~glamm/ical/>
- ZSREP is the Zaurus Synchronization Reverse Engineering Project. <http://www.cyph.org/zsrep/>
- Asta Technology Group, Inc. introduces the SkyWire™ - Smart Client Toolkit a group of cross platform applications and libraries, designed to allow you to manipulate your Enterprise data from your Palm, PocketPC or Win32 Client securely and get data that is accurate to the second it is requested. <http://www.astaskywire.com/products/skywire/>
- Powerful combination of a state-of-the-art, 32-bit Pascal compiler and an easy-to-use, adaptable integrated development environment that allows you to quickly build sophisticated applications for the PalmOS <http://www.winsoft.sk/pstudio.htm>
- mobile access to personal information, Wireless synchronization of PIM data and email in all devices that support the open SyncML standard.
<http://www.space2go.de/index.jsp?language=ctLng0>
- Service nur für ONE Mobile Office for Microsoft Exchange und ONE Mobile Office for Lotus Domino registrierte User
<https://office.one.at/intra.htm>
- empty PHP Sync project, I wrote a message to the author on 2003-09-30
<http://sourceforge.net/projects/phpsyncml/>
- general SyncML Resources <http://sync4j.sourceforge.net/web/resources.html>
- Open source options for implementing the protocol <http://www-106.ibm.com/developerworks/xml/library/x-syncml3.html>
- SyncML C Reference Toolkit. The purpose of this toolkit is to provide a reference implementation of the SyncML protocols. The toolkit is designed for both Clients and Servers, for a variety of platforms.
<http://sourceforge.net/projects/syncml-ctoolkit/>
- Get Mirko Mrowczynski's example SyncML application which is in German and utilizes the SyncML reference toolkit (RTK). <http://www-106.ibm.com/developerworks/xml/library/x-syncml3.html>
- LibSyncML Project <http://libsyncml.sourceforge.net/>
- Syncing with Horde Applications <http://horde.org/sync/>
- SyncML support for Ogo <http://www.opengroupware.org/en/projects/syncml/>
- Synthesis SyncML Server Version 2.0, SyncML Clients PRO for PalmOS and PocketPC
<http://www.synthesis.ch/>
- PHP iCalendar <http://phpicalendar.sourceforge.net/nuke/>
- MultiSync is a free modular program to synchronize calendars, addressbooks and other PIM data between programs on your computer and other computers, mobile devices, PDAs or cell phones. MultiSync works on any Gnome platform, such as Linux. MultiSync supports several different data formats and sources including SyncML and PalmOS
<http://www.multisync.org/index.shtml>
- <http://spspace.com>, Magically SyncML Client for Outlook. Supports only synchronization with there own server
- http://www.weblicon.net/html/products_syncml.html
SyncML client for Outlook, Palm and PocketPC

4.8. SyncML capable devices

- complete list http://www.openmobilealliance.org/syncml/compliant_products.htm

Handys

Nokia

- Modell N-Gage
- Modell 3100
- Modell 3108
- Modell 3200
- Modell 3300
- Modell 3310
- Modell 3330
- Modell 3510
- Modell 3650 (über download von Website)
- Modell 3660
- Modell 5100
- Modell 5140
- Modell 6100
- Modell 6220
- Modell 6230
- Modell 6310i
- Modell 6600
- Modell 6800
- Modell 6820
- Modell 7200
- Modell 7250
- Modell 7250i
- Modell 7600
- Modell 7650 (über download von Website)
- Modell 7700
- Modell 8850
- Modell 8910
- Modell 8910i
- Modell 9210
- Modell 9210i

Sony Ericsson

- Modell Z600
- Modell Z200
- Modell T610
- Modell P900
- Modell T630
- Modell Z1010
- Modell P800
- Modell T68i

- Modell T68m
- Modell T68
- Modell T65
- Modell T39m
- Modell R520m
- Modell T100
- Modell T200
- Modell T300
- Modell T310
- Modell T600

Alcatel

- Modell One Touch 715
- Modell One Touch 735
- Modell One Touch 535

Beaucom

- Modell Enigma

E-Plus

- Modell Hiptop

Haier

- Modell P5

LG-Electronics

- Modell LG-G7050

Motorola

- Modell T720
- Modell A760
- Modell A830
- Modell A835
- Modell A920
- Modell A925
- Modell C450
- Modell E380
- Modell MPx200
- Modell T722i
- Modell V300
- Modell V525
- Modell V600

NEC

- Modell e525
- Modell e606
- Modell e808
- Modell n21i
- Modell n223i
- Modell n22i
- Modell n31i
- Modell n341i

Sendo

- Modell X
- Modell J530
- Modell M550

Siemens

- Modell A52
- Modell A55
- Modell A60
- Modell C55
- Modell C62
- Modell CL50
- Modell M50
- Modell MC60
- Modell ME45
- Modell S45
- Modell S45i
- Modell SL45i
- Modell ST55
- Modell S55
- Modell SX1
- Modell SX45
- Modell M55
- Modell SL55
- Modell U15

O2

- Modell X1
- Modell XDA
- Modell XDA II

Panasonic

- Modell GD 96
- Modell GD 87

Philips

- Modell Fisio 825

RIM

- Modell Blackberry 7230
- Modell Blackberry 6230

Sagem

- Modell my3088
- Modell myV-65
- Modell myX-5

Samsung

- Modell SGH-A800

- Modell SGH-C100
- Modell SGH-D700
- Modell SGH-E100
- Modell SGH-E700
- Modell SGH-R710
- Modell SGH-P400
- Modell SGH-P410
- Modell SGH-S300
- Modell SGH-S300M
- Modell SGH-S500
- Modell SGH-V200
- Modell SGH-X100
- Modell SGH-X600

Sharp

- Modell GX1

- Modell GX10
- Modell GX10i
- Modell GX20

T-Mobile

- Modell MDA
- Modell MDA II

Tel.Me

- Modell T909c
- Modell T910
- Modell T919

Toshiba

- Modell TS21i

4.9. Misc SyncML: <http://www.nexthaus.com>

- SyncJe™ Client for Palm OS
- SyncJe™ Conduit for Palm OS
- SyncJe™ Client for Pocket PC
- SyncJe™ Client for Outlook
- SyncJe™ Personal Server for Outlook

Nexthaus seems to have interesting products, but our test with Client for Outlook shows some bugs in SyncML implementation. We are evaluating further.

The product Conduit for Palm OS seems to have another problem: It generates a new IEMSI (aka LcoalURI) every time the application is started. This prevents everything going behind making garbage.

5. Installation

Misc unsorted things:

- to use the XML-RPC interface (./xmlrpc.php) you must enable db session handling (php4 session handling does not work with xml-rpc ATM)
- XML RPC needs the PHP XML-RPC extension installed. Uncomment extension=php_xmlrpc.dll in php.ini
- see document Installation Guide for Sync4j.swx/.pdf for more information

6. Glossary

Soft deletion and hard deletion

There also has to be a special handling of the delete operation. Not every client has enough space for every database item. Sometimes we want to remove an item from just one client, but not from the others. This operation is called a soft delete (deletion on one device) as opposed to a hard delete (deletion on all devices).

One way would be for the client to keep track of the phased out items. But this would not solve the problem: The client would still have to know about all records it tried not to know about. Therefore a soft delete has to be handled internally on the server:

The server keeps the record that this entry is invisible to a particular client. (by Maximilian Berger)

Slow Sync

A regular sync is only possible when both databases were synchronized before and when both have their transaction logs or are able to recreate them. If this is not possible they have to initiate a so called “slow sync”.

1. Milestones

Milestone 1: (8 weeks, 19.01.2004, reached on 13.01.2004)

- XML-RPC-Interface for Sync4j (CW)
- Interface declarations and implementation with XML-RPC for authentications layer 4 – 5 (ML)
- Layer 3 DB-setup and authentication (MK)
- Goal: a device may authorize on phpGroupware

Milestone 2: (6 weeks, 02.02.2004, reached on 26.01.2004)

- one-way sync from client only (phpGW -> P800)

Milestone 3: (3 weeks, 23.02.2004, reached on 04.03.2004)

- Full (slow) sync with jotter/notes/memo (text/plain) application type
- remove Sync4j database usage
- start adding internal mapping

Milestone CeBit: 18.03.2004 – 24.03.2004

- show sync capabilities with SyncML client

Milestone 4: (4 weeks, 22.03.2004)

- two-way sync (including conflict handling + time stamp and dirty flag handling)

Milestone 5: (8 weeks, 17.05.2004)

- GUI for layer 3 (phpGW sync module – get users wish to solve conflicts)

Milestone 6: (2 weeks, 31.05.2004)

- implementation additional layer one modules
- add support for more than the preferred mimetypes

Milestone 7: (8 weeks, 26.07.2004)

- add support for non SyncML clients like Palm

Milestone 8: (12 weeks, 18.10.2004)

- replace sync4j with own SyncML implementation written in PHP

1. History

<i>Day</i>	<i>Changed</i>
20.11.03	<ul style="list-style-type: none">• improved description of milestones• first public version
22.12.03	<ul style="list-style-type: none">• updated with comments from the maillinglist and from the meeting with Dave Hall on probusiness, Hannover
18.12.03	<ul style="list-style-type: none">• layer 4 is renamed to SyncModule IPC and uses the new phpGW IPC interface now
13.01.04	<ul style="list-style-type: none">• we are on Milestone 1 as of today, misc updates
26.01.04	<ul style="list-style-type: none">• added chapter 4.8, SyncML device list• we are on Milestone 2 as of today
31.01.04	<ul style="list-style-type: none">• changed database description to reflect actual development
09.02.04	<ul style="list-style-type: none">• Updated misc parts of documentation, more links and tests for SyncML software
16.02.04	<ul style="list-style-type: none">• Updated database description
17.02.04	<ul style="list-style-type: none">• Added list of SyncML capable devices
04.03.04	<ul style="list-style-type: none">• Annonced Milestone 3
09.03.04	<ul style="list-style-type: none">• Additional database field MimeVersion in SyncSources
01.05.04	<ul style="list-style-type: none">• Database changes for conflict handling
19.05.04	<ul style="list-style-type: none">• Complete proofreading of the whole document, adaption to nearly all chapters to reflect current implementation details• Annonced Milestone 5,