Sync4j

SyncConnector Exchange 1.2 Admin Guide

Table of Contents

1. Introduction **3**
1.1. Comments and Feedbacks 3
2. Architecture **4**
2.1. The Big Picture 4
2.2. SyncServer – Exchange Server Communication 4
2.3. Security 5
2.3.1. SyncServer Authentication 5
2.4. ExchangeSyncSource Type 5
2.4.1. WebDAV 5
3. Installation and Administration **7**
3.1. SyncConnector Exchange Registration 7
3.2. Configuration 7
3.2.1. Configuration examples 8
Exchange Contacts SyncSource 8
Exchange Calendar SyncSource 8
4. Internal design **9**
4.1. Overview 9
4.2. Items Identification 10
4.3. Detecting and Applying Item Changes 11
4.4. Local Cache Database 11
4.5. Error Handling 12
4.6. SyncConnector – Exchange dialog examples 12
4.6.1. Adding a contact: 12
4.6.2. Updating a Contact 12
4.6.3. Listing Contacts 13
4.6.4. Deleting a Contact 13
4.6.5. Response to a Successful Addition 13
4.6.6. Response to a Successful Update 14
4.6.7. Response to a Failed Addition (not Existing URN) 14
4.6.8. Response to a Successfully Deletion 14
4.6.9. Response to a Failed Deletion (Unexisting Contact) 15
4.6.10. Adding an Appointment 15
4.6.11. Updating an Appointment 15
4.6.12. Listing Appointements 16
4.6.13. Deleting an Appointement 16
4.6.14. Selecting Appointment Items Modified after a Given Date 16
5. Sync4j Licensing **18**
6. Appendices **19**
6.1. References 19
6.2. Contact Properties 19
6.3. Calendar properties 20
6.4. Task properties 21
6.5. Note properties 21

1. Introduction

This document is intended for developers and administrators who have to manage Sync4j SyncConnector Exchange 1.2.x. It includes:

- Architectural Overview
- Configuration and Administration

1.1. Comments and Feedbacks

The Sync4j Project team wants to hear from you! Please submit your questions, comments, feedbacks or testimonials to sync4j-users@lists.sourceforge.net.

2. Architecture

This section describes the architecture of the SyncConnector Exchange.

2.1. The Big Picture

The system architecture of the SyncConnector Exchange is drown in Figure 1.

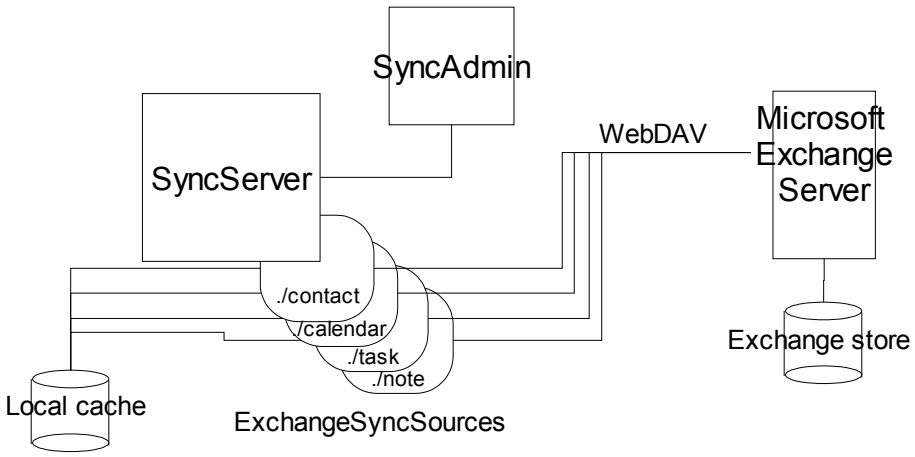

Figure 1 - Sync4j SyncConnector Exchange high level architecture

PIM data are stored in the Microsoft Exchange store and are managed by the Microsoft Exchange Server. When SyncServer receives a synchronization request from a client addressed to one of the ExchangeSyncSource, the Local cache is queried for latest updates and, if any item has been updated client side or server side, the ExchangeSyncSource talks to the Exchange server through the WebDAV protocol.

SyncConnector Exchange is administered with a dedicated plug-in for the SyncServer administration tool *SyncAdmin* by which an administrator can create/modify/delete ExchangeSyncSources.

2.2. SyncServer – Exchange Server Communication

The remote access protocol recommended by Microsoft in a distributed environment is WebDAV.

The WebDAV protocol is an extension to HTTP that you can use to build Web applications that are writable. Using WebDAV protocol methods, you can create, copy, delete, move, or search for resources in the Exchange store as well as set and search for resource properties.

The RFC 2518 specification defines extensions to the Hypertext Transfer Protocol 1.1 (HTTP/1.1), which is defined in RFC 2616. The extensions include a new set of protocol methods and request headers used to move, copy, delete, and create collections of resources. Additionally, clients can set and get properties about resources along with the resource content itself.

The extensions relating to getting and setting properties require a rich format for data transfer. RFC 2518 defines the XML 1.0 as the format for wire data transfer. Additionally, a namespace in XML is required to provide scope for property names.

For more information about WebDAV see [2] and [3].

2.3. Security

Being WebDAV an HTTP based protocol, security and authentication is based on HTTP security and authentication.

NOTE: a standard Microsoft Exchange installation includes the WebDAV interface, which is based on IIS as web server. By default, IIS is configured to use the proprietary authentication protocol NTLM, which is implemented only in Internet Explorer. Since the ExchangeSyncSource uses raw HTTP requests to the Exchange server, IIS must be configured to use basic authentication. This is done through the Exchange System Manager, changing the options available on the panel *Admistrative Groups / First Administrative Group / Servers / [server] / Protocols / Http / Exchange Virtual Server / Exchange*.

Authentication is also the mechanism used by Exchange server to support multiuser. When a WebDAV request is served, it allows to operate only with data belonging to the requesting user.

2.3.1. SyncServer Authentication

Because of the described security model, the SyncServer should skip its own authentication done via properties files or database. This is easily done configuring SyncServer to use sync4j.framework.security.SimpleOfficier instead of the default DBOfficier.
In the case of specific needs, different authentication mechanisms can be implemented developing a custom javax.security.auth.spi.LoginModule.

2.4. ExchangeSyncSource Type

The ExchangeSyncSource type is the mean the SyncServer interacts with the Exchange store. To do so, the sync source implements the needed WebDAV requests and interprets the received responses.

Because of differences in item handling between different item types (contacts, events, tasks, notes), four versions of ExchangeSyncSource type are provided:

- sync4j.exchange.engine.source.ExchangeContactSyncSource
- sync4j.exchange.engine.source.ExchangeCalendarSyncSource
- sync4j.exchange.engine.source.ExchangeTaskSyncSource
- sync4j.exchange.engine.source.ExchangeNoteSyncSource

All those classes, like any sync source, implement the *sync4j.framework.engine.source.SyncSource* interface and extend *sync4j.exchange.engine.source.ExchangeSyncSource*.

2.4.1. WebDAV

The Exchange store is a container of items, where each single item can, in turn, contains other items (in the case of folder items) or properties (in the case of data items).

Properties items can be queried and modified by a remote client through the WebDAV protocol. This is based on an extension of the HTTP protocol as transport protocol and on an XML based RPC protocol by which it is possible to query and modify items and properties.

The Exchange store is based on the item types shown in Figure 2. In this figure, item types are represented as groups of properties identified by an XML namespace.

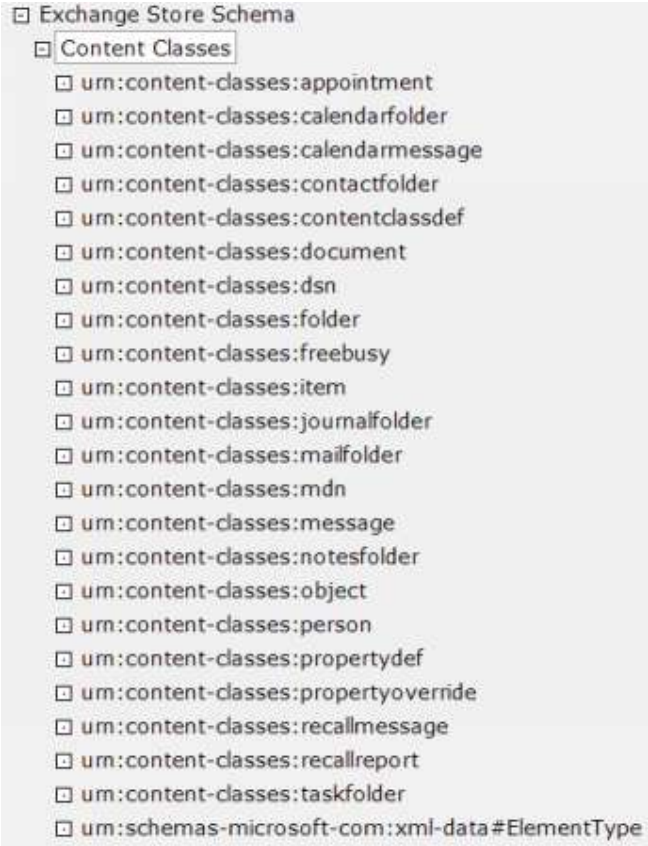

Figure 2 - Exchange store schema

The full list of the properties for each item type (contact, appointment, task and note) is available in Appendix A.

3. Installation and Administration

The SyncConnector Exchange is administered and configured through the SyncAdmin interface. In this section, it is presented how to set up the SyncServer database in order to have the connector configured and the panel used to create and configure new sync sources based on the registered sync source types.

3.1. SyncConnector Exchange Registration

The connector is distributed as a Sync4j module (see [6]); the archive name is syncconnectorexchange-1.2.zip.

After installing the module as describe in the administration guide[7], opening the SyncAdmin you should have the new SyncConnector displayed. Expanding the children nodes you should have the tree shown in Figure 3.

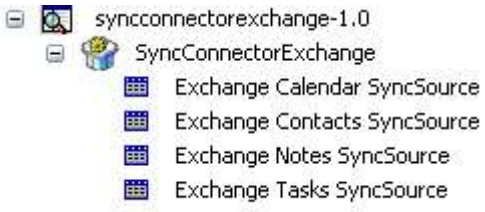

Figure 3 - SyncConnector Exchange registration

3.2. Configuration

A ExchangeSyncSource is configued with the properties listed in the table below.

Property	***Description***
Source URI	The sync source URI
Exchange folder	exchange Server Name / exchange item folder e.g. Exchange/Contacts
Name	The SyncSource name.
Type	The Data content mime type.
Host	The Exchange server hostname/ip address.
Port	The Exchange server port.
Supported Types	The Data content mime type supported
Supported Version	The Data content mime type version supported

Property	Description
Encode	Should the Data content be Base64 encoded?
Conversion Type	XML / VCard / ICal

3.2.1. Configuration examples

Exchange Contacts SyncSource

Property	Value
Source URI	./contacts
Exchange folder	Exchange/Contacts
Name	contacts
Type	text/x-vcard
Host	exchange.sync4j.org
Port	80
Supported Types	text/x-vcard
Supported Version	1.0
Encode	false
Conversion Type	VCard

Exchange Calendar SyncSource

Property	Value
Source URI	./calendars
Exchange folder	Exchange/Calendar
Name	calendars
Type	text/x-vcalendar
Host	exchange.sync4j.org
Port	80
Supported Types	text/x-vcalendar
Supported Version	1.0
Encode	false
Conversion Type	ICal

4. Internal design

The following sections describe how SyncConnector Exchange manages internal data and the algorithm for data synchronization. This information is not meant for administrator but for developers wanting to extend the capabilities of SyncConnector Exchange.

4.1. Overview

SyncConnector Exchange is composed of the logical modules shown in the collaboration diagram of Figure 4.

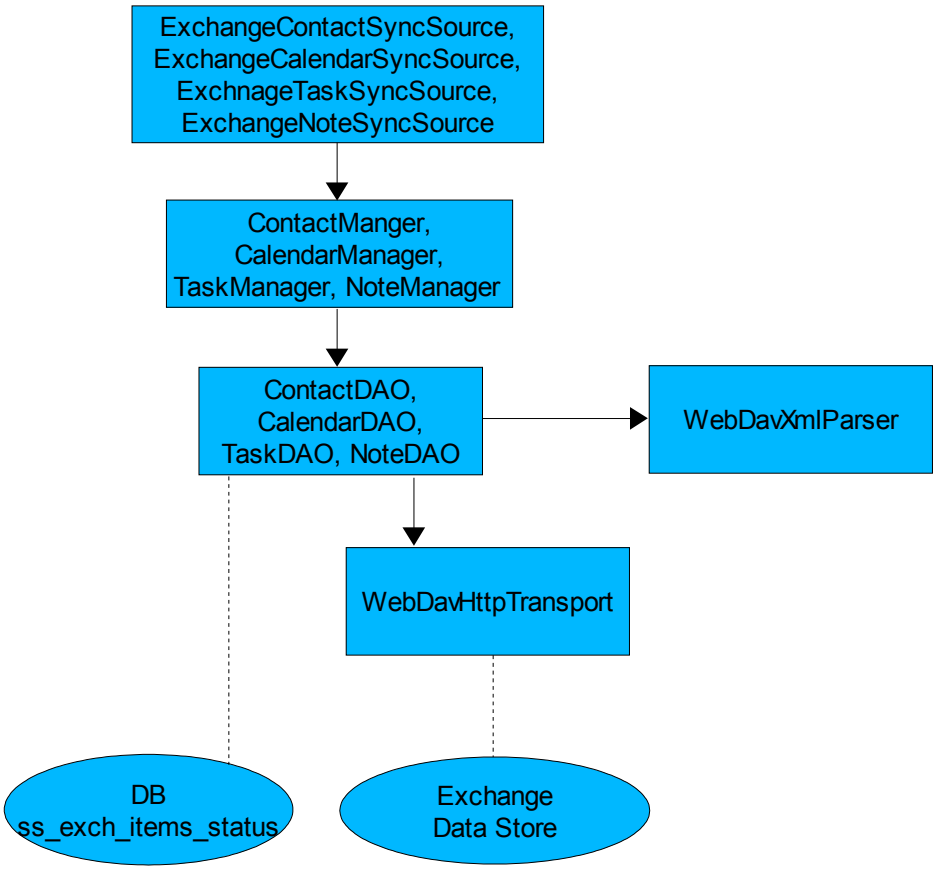

Figure 4 - SyncConnector Exchange collaboration diagram

The table below describes the role of each group of classes.

ExchangeContactSyncSource, ExchangeCalendarSyncSource, ExchnageTaskSyncSource, ExchangeNoteSyncSource	This are the components to be plugged into the synchronization engine. They are implementation of the sync4j.framework.engine.source.SyncSource interface. These classes make use of the corresponding manager classes of the above layer in order to access the data layer. Not displayed by the figure, sync source and manager objects deal with the item data by the mean of the model classes Event, Contact, Task, Note.
CalendarManager, ContactManager, TaskManager, NoteManager	This classes create an interface layer on top of the data access layer and include the logic to retrieve the data associated to the items and to combine them in the model classes.
CalendarDAO, ContactDAO, TaskDAO, NoteDAO	This layer abstracts the access to the datasource (database for items status and Exchange store). The WebDAV XML messages returned by the Exchange server are parsed by WebDavXmlParser.
WebDavXmlParser	Is the class that interprets the WebDAV messages obtained by the server.
WebDavHttpTransport	This is the class that implements the connection logic to the Exchange server.

4.2. Items Identification

Any item in the Exchange store is addressed by a resource address and identified by a user-id. The resource address identifies an item by its name and position. For example /contact/home/John Brown.eml addresses a contact stored in a resource called John Brown.eml under the home subfolder.
The resource address is used mainly for update and delete operations, since they require to specify which resource is involved.

However, the resource url cannot be considered a long-term identification key because an item can be moved under different folders. For the purpose of uniquely identify an object in the Exchange store, the property repl-uid is used as it is associated to any item in the store.

The issue of creating new items in the Exchange store is resolved constructing a resource url as follows:

/<user>/<source uri>/<now timestamp>.eml

The item repl-uid assigned by Exchange is returned in the WebDAV response header as in the following example:

```
HTTP/1.1 207 Multi-Status
Server: Microsoft-IIS/5.0
Date: Tue, 20 Jan 2004 15:37:02 GMT
MS-Exchange-Permanent-URL:              http://192.168.0.10/Exchange/fabio/-
FlatUrlSpace-/5eb
c95ae2cdf504c98296172915e02d2-5d/5ebc95ae2cdf504c98296172915e02d2-3e26
Repl-UID: <rid:5ebc95ae2cdf504c98296172915e02d2000000003e26>
Content-Type: text/xml
Content-Length: 436
ResourceTag:
<rt:5ebc95ae2cdf504c98296172915e02d2000000003e265ebc95ae2cdf504c982
96172915e02d2000000004dbb>
MS-WebStorage: 6.0.6249
```

4.3. Detecting and Applying Item Changes

Two important properties stored along with an item in the Exchange store are `<a:creationdate>` and `<a:getlastmodified>` whose return respectively the item creation date and the item last modification timestamp. These properties are important when a fast sync is performed, since the sync source should query for changes happened since the last synchronization. However, when an item is physically deleted from the Exchange store, it disappears from the system and then there is no way to retrieve it or retrieve when it was deleted.
Therefore, a local items cache is needed in order keep track of the existing items at the time of the last synchronization. We call this database the *Local cache*.

The algorithm for changes detection is the following:

1. read all items repl-uid, resourceName, creationDate, lastModified from the Exchange store and put them in exchangeItems
2. read all items repl-id, lastModified, from the local cache and put them in localItems
3. foreach item i in exchangeItems
    1. if i is in localItems
        1. if i.lastModified != localItems (i) lastModified
            1. i is updated
        2. end if
    2. else
        1. i is new
    3. end if
4. foreach item i in localItems
    1. if i is not in exchangeItems
        1. i is deleted
    2. endif
5. endforeach

This procedure is executed in the ExchangeSyncSource's beginSync() method if the sync mode is one of the fast sync modes. The retrieved new, updated and deleted items are temporarily stored until the corresponding getNew/Updated/DeletedSyncItem() methods are called. When either getNewSyncItems() or getUpdatedSyncItems() are called by the sync engine, the corresponding items content is fetched by the Exchange store with proper WebDAV calls. All items must be returned in a single call.
Note that in the comparison steps, uids are compared and not resource urls.

If the sync mode is one of the slow sync modes, no step in the algorithm is done. When getAllSyncItem() is called by the sync engine, the items properties are fetched directly from the Exchange store, possibly in one single WebDAV call.

If the client sends items to add to the Exchange store, the new resource is created and the returned uid is added to the local cache.
In the case of client updates or deletes, the uid is used to look up the resource address in the exchangeItems list so that the obtained address can be used in the WebDAV call.

In the sync source's endSync() method read from Exchange Server the existing items, and are stored in the local cache (after cleaning it).

4.4. Local Cache Database

The local cache database is composed of the following tables:

SS_EXCH_ITEM_STATUS			
id	varchar(200)	primary key	Item unique identifier (repl-uid) in the syncsource specified by uri.
uri	varchar(128)	primary key	SyncSource uri.
principal	varchar(10)	index	Item owner.

SS_EXCH_ITEM_STATUS			
last_modified	timestamp		item last modified on Exchange Server

Note that items ids are stored with their owner, who is the principal associated to the user who is performing the sync.

4.5. Error Handling

Any error should happen in the sync source results in a SyncSourceException being thrown. The error condition is logged before throwing the exception and optionally the cause can be set to the originating exception.

4.6. SyncConnector – Exchange dialog examples

The following sections show some examples on how SyncConnector Exchange interacts with Exchange server through WebDAV.

4.6.1. Adding a contact:

```
PROPPATCH /Exchange/fabio/contacts/luigiafassina.eml HTTP/1.1
Host: 192.168.0.10
Content-Type: text/xml
Content-Length: 741
Authorization: Basic ZmFiaW86ZmFiaW8x

<?xml version="1.0" encoding="utf-8" ?>
<D:propertyupdate                                                    xmlns:D="DAV:"
xmlns:EX="http://schemas.microsoft.com/exchange/"
xmlns:HM="urn:schemas:contacts:">
  <D:set>
    <D:prop>
      <D:contentclass>urn:content-classes:person</D:contentclass>
      <EX:outlookmessageclass>IPM.Contact</EX:outlookmessageclass>
      <HM:cn>Luigia Fassina</HM:cn>
      <HM:nickname>Lu</HM:nickname>
      <HM:email1>luigia.fassina@sync4j.org</HM:email1>
      <HM:title>Luigia Fassina</HM:title>
      <HM:fileas>luigiafassina</HM:fileas>
      <HM:givenName>Luigia</HM:givenName>
    </D:prop>
  </D:set>
</D:propertyupdate>
```

luigiafassina.eml on the first line is the key that uniquely identifies the item.

4.6.2. Updating a Contact

```
PROPPATCH /Exchange/fabio/contacts/luigiafassina.eml HTTP/1.1
Host: 192.168.0.10
Content-Type: text/xml
Content-Length: 741
Authorization: Basic ZmFiaW86ZmFiaW8x

<?xml version="1.0" encoding="utf-8" ?>
<D:propertyupdate                                                    xmlns:D="DAV:"
xmlns:EX="http://schemas.microsoft.com/exchange/"
xmlns:HM="urn:schemas:contacts:">
  <D:set>
    <D:prop>
      <D:contentclass>urn:content-classes:person</D:contentclass>
      <EX:outlookmessageclass>IPM.Contact</EX:outlookmessageclass>
```

```
      <HM:cn>Luigia Fassina</HM:cn>
      <HM:nickname>Lu</HM:nickname>
      <HM:email1>lu@sync4j.org</HM:email1>
      <HM:title>Luigia Fassina</HM:title>
      <HM:fileas>luigiafassina</HM:fileas>
      <HM:givenName>Luigia</HM:givenName>
    </D:prop>
  </D:set>
</D:propertyupdate>
```

4.6.3. Listing Contacts

```
PROPFIND /Exchange/fabio/contacts HTTP/1.1
Host: 192.168.0.10
Depth: 1,noroot
Range: rows=0-9
Content-Type: text/xml
Content-Length: 741
Authorization: Basic ZmFiaW86ZmFiaW8x

<?xml version="1.0" encoding="utf-8" ?>
<D:propfind                                                    xmlns:D="DAV:"
xmlns:EX="http://schemas.microsoft.com/exchange/"
xmlns:HM="urn:schemas:contacts:">
<D:allprop/>
</D:propfind>
```

This example returns all fabio's contacts.

4.6.4. Deleting a Contact

```
DELETE /Exchange/fabio/contacts/luigiafassina.eml HTTP/1.1
Host: 192.168.0.10
Content-Length: 50
Authorization: Basic ZmFiaW86ZmFiaW8x
```

4.6.5. Response to a Successful Addition

```
HTTP/1.1 207 Multi-Status
Server: Microsoft-IIS/5.0
Date: Tue, 20 Jan 2004 15:31:28 GMT
MS-Exchange-Permanent-URL:              http://192.168.0.10/Exchange/fabio/-
FlatUrlSpace-/5eb
c95ae2cdf504c98296172915e02d2-5d/5ebc95ae2cdf504c98296172915e02d2-3e26
Repl-UID: <rid:5ebc95ae2cdf504c98296172915e02d2000000003e26>
Content-Type: text/xml
Content-Length: 477
ResourceTag:
<rt:5ebc95ae2cdf504c98296172915e02d2000000003e265ebc95ae2cdf504c982
96172915e02d2000000004db6>
MS-WebStorage: 6.0.6249


<?xml                                          version="1.0"?><a:multistatus
xmlns:b="http://schemas.microsoft.com/exchange/"
xmlns:c="urn:schemas:contacts:" xmlns:a="DAV:">
<a:response>
<a:href>http://192.168.0.10/Exchange/fabio/Contacts/Johnniepippo1.eml</a:hr
ef>
<a:status>HTTP/1.1 201 Created</a:status><a:propstat><a:status>HTTP/1.1 200
OK</a:status>
<a:prop>
```

```
 <a:contentclass/><b:outlookmessageclass/><c:cn/><c:nickname/><c:email1/><c
:title/><c:fileas/><c:givenName/></a:prop></a:propstat></a:response></a:mul
tistatus>
```

The operation status is specified by the line:

```
<a:status>HTTP/1.1 201 Created</a:status>
```

4.6.6. Response to a Successful Update

```
HTTP/1.1 207 Multi-Status
Server: Microsoft-IIS/5.0
Date: Tue, 20 Jan 2004 15:37:02 GMT
MS-Exchange-Permanent-URL:              http://192.168.0.10/Exchange/fabio/-
FlatUrlSpace-/5eb
c95ae2cdf504c98296172915e02d2-5d/5ebc95ae2cdf504c98296172915e02d2-3e26
Repl-UID: <rid:5ebc95ae2cdf504c98296172915e02d2000000003e26>
Content-Type: text/xml
Content-Length: 436
ResourceTag:
<rt:5ebc95ae2cdf504c98296172915e02d2000000003e265ebc95ae2cdf504c982
96172915e02d2000000004dbb>
MS-WebStorage: 6.0.6249


<?xml                                          version="1.0"?><a:multistatus
xmlns:b="http://schemas.microsoft.com/exchan
ge/"                                        xmlns:c="urn:schemas:contacts:"
xmlns:a="DAV:"><a:response><a:href>http://1
92.168.0.10/Exchange/fabio/Contacts/Johnniepippo1.eml</a:href><a:propstat><
a:sta
tus>HTTP/1.1                                                              200
OK</a:status><a:prop><a:contentclass/><b:outlookmessageclass/><
c:cn/><c:nickname/><c:email1/><c:title/><c:fileas/><c:givenName/></a:prop><
/a:pr
opstat></a:response></a:multistatus>
```

4.6.7. Response to a Failed Addition (not Existing URN)

```
HTTP/1.1 409 Conflict
Server: Microsoft-IIS/5.0
Date: Tue, 20 Jan 2004 15:41:51 GMT
Connection: close
Content-Type: text/html
Content-Length: 140
MS-WebStorage: 6.0.6249

<body><h2>HTTP/1.1 409 Conflict</h2></body></HTML>; charset=utf-8">
```

4.6.8. Response to a Successfully Deletion

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Tue, 20 Jan 2004 15:45:43 GMT
Content-Length: 0
ResourceTag: <rtd:5ebc95ae2cdf504c98296172915e02d2000000003e26>
MS-WebStorage: 6.0.6249
```

4.6.9. Response to a Failed Deletion (Unexisting Contact)

```
HTTP/1.1 404 Resource Not Found
```

```
Server: Microsoft-IIS/5.0
Date: Tue, 20 Jan 2004 15:43:54 GMT
Connection: close
Content-Type: text/html
Content-Length: 4040
MS-WebStorage: 6.0.6249

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<html dir=ltr>

<head>
<style>
a:link                  {font:8pt/11pt verdana; color:FF0000}
a:visited               {font:8pt/11pt verdana; color:#4e4e4e}
</style>

<META NAME="ROBOTS" CONTENT="NOINDEX">

<title>The page cannot be found</title>

L'errore indicato è:
HTTP/1.1 404 Resource Not Found
```

4.6.10. Adding an Appointment

```
PROPPATCH /Exchange/fabio/calendar/calendarprova.eml HTTP/1.1
Host: 192.168.0.10
Content-Type: text/xml
Content-Length: 741
Authorization: Basic ZmFiaW86ZmFiaW8x

<?xml version="1.0" encoding="utf-8" ?>
<D:propertyupdate                                                xmlns:D="DAV:"
xmlns:EX="http://schemas.microsoft.com/exchange/"
xmlns:HM="urn:schemas:calendar:" xmlns:e="urn:schemas:httpmail:">
  <D:set>
    <D:prop>
      <D:contentclass>urn:content-classes:appointment</D:contentclass>
      <EX:outlookmessageclass>IPM.Appointment</EX:outlookmessageclass>
      <e:subject>oggi telefonare al signor verri</e:subject>
      <HM:location>meetappt Location</HM:location>
      <HM:dtstart>2004-01-10T22:00:00.000Z</HM:dtstart>
      <HM:dtend>2004-01-10T22:30:00.000Z</HM:dtend>
    </D:prop>
  </D:set>
</D:propertyupdate>
```

4.6.11. Updating an Appointment

```
PROPPATCH /Exchange/fabio/calendar/calendarprova.eml HTTP/1.1
Host: 192.168.0.10
Content-Type: text/xml
Content-Length: 741
Authorization: Basic ZmFiaW86ZmFiaW8x

<?xml version="1.0" encoding="utf-8" ?>
<D:propertyupdate                                                xmlns:D="DAV:"
xmlns:EX="http://schemas.microsoft.com/exchange/"
xmlns:HM="urn:schemas:calendar:" xmlns:e="urn:schemas:httpmail:">
  <D:set>
```

```
    <D:prop>
      <D:contentclass>urn:content-classes:appointment</D:contentclass>
      <EX:outlookmessageclass>IPM.Appointment</EX:outlookmessageclass>
      <e:subject>oggi telefonare al signor rossi</e:subject>
      <HM:location>meetappt Location</HM:location>
      <HM:dtstart>2004-01-10T22:00:00.000Z</HM:dtstart>
      <HM:dtend>2004-01-10T22:30:00.000Z</HM:dtend>
    </D:prop>
  </D:set>
</D:propertyupdate>
```

4.6.12. Listing Appointements

```
PROPFIND /Exchange/fabio/calendar HTTP/1.1
Host: 192.168.0.10
Depth: 1,noroot
Range: rows=0-9
Content-Type: text/xml
Content-Length: 741
Authorization: Basic ZmFiaW86ZmFiaW8x

<?xml version="1.0" encoding="utf-8" ?>
<D:propfind                                                    xmlns:D="DAV:"
xmlns:EX="http://schemas.microsoft.com/exchange/"
xmlns:HM="urn:schemas:calendars:">
<D:allprop/>
</D:propfind>
```

4.6.13. Deleting an Appointement

```
DELETE /Exchange/fabio/calendar/calendarprova.eml HTTP/1.1
Host: 192.168.0.10
Content-Length: 50
Authorization: Basic ZmFiaW86ZmFiaW8x
```

4.6.14. Selecting Appointment Items Modified after a Given Date

WebDAV defines an SQL-like language through which execute generic queries on the Exchange store.

```
SEARCH /Exchange/fabio/calendar HTTP/1.1
Host: 192.168.0.10
Depth: 1,noroot
Range: rows=0-7
Content-Type: text/xml
Content-Length: 741
Authorization: Basic ZmFiaW86ZmFiaW8x

<?xml version="1.0" encoding="utf-8" ?>
<D:searchrequest xmlns:D = "DAV:" >
   <D:sql>
                Select   "urn:schemas:calendar:location"   AS   location,
      "urn:schemas:httpmail:subject"         AS         subject         FROM
"/Exchange/fabio/calendar"    WHERE   "D:getlastmodified"   &gt;   '1999/09/27
00:00:00'
   </D:sql>
</D:searchrequest>
```

5. Sync4j Licensing

Sync4j has two licensing options, following what is known as "dual licensing" model. After the adoption by MySQL, this model is becoming very successful as a way to support open source development.
The guiding business principle of dual licensing is one of fair exchange, or Quid pro Quo ("something for something"). From a licensing perspective, there are two different products depending on usage and distribution, though technically they have the same source code.

- For those developing open source applications, the Open Source License allows you to offer your software under an open source / free software license (GLP) to all who wish to use, modify, and distribute it freely. The Open Source License allows you to use the software at no charge under the condition that if you use Sync4j in an application you redistribute, the complete source code for your application must be available and freely redistributable under reasonable conditions. Sync4j bases its interpretation of the GPL on the Free Software Foundation's Frequently Asked Questions.
- The Commercial License, which allows you to provide commercial software licenses to your customers or distribute Sync4j-based applications within your organization. This is for organizations that do not want to release the source code for their applications as open source / free software; in other words they do not want to comply with the GNU General Public License (GPL). If you want more information on pricing, please contact license@sync4j.org.

The idea is to get the best out open source (high quality software, a community of people working together, no vendor lock-in), while providing a source of income to pay for the development of the software (yep, Sync4j developers need to eat...).

In their simplest form, the following are general licensing guidelines:

- If your software is licensed under either the GPL-compatible Free Software License as defined by the Free Software Foundation or approved by OSI, then use our GPL licensed version.
- If you distribute a proprietary application in any way, and you are not licensing and distributing your source code under GPL, you need to purchase a commercial license of Sync4j

Commercially licensed customers get commercially supported product with assurances from Funambol. Commercially licensed users are also free from the requirement of making their own application open source.

For OEM's, ISVs, corporate, and government users, a commercial license is the proper solution because it provides you with assurance from the vendor and releases you from the strict requirements of the GPL license.

Nevertheless, you can test Sync4j under the GPL license and inspect the source code before you purchase a commercial non-GPL license.

6. Appendices

6.1. References

[1] *SyncConnector Exchange Requirements Document*, October 2003
[2] http://www.webdav.org
[3] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wss/wss/wss_references_webdav.asp
[4] Quartz Enterprise Job Scheduler, http://www.quartzscheduler.org
[5] JBoss Administration and Development, Second Edition
[6] Sync4j Modules Development Guide, Sync4j, February 2004
[7] Sync4j SyncServer 4.0.x Administration Guide, November 2004

6.2. Contact Properties

The following properties are associated to an Exchange contact item and supported by the connector:

General	
First name	<FirstName>
Last name	<LastName>
Middle name	<MiddleName>
Nick name	<NickName>
Title	<Title>
Company	<CompanyName>
Spouse	<Spouse>
File as	<FileAs>
Home address	
Home street	<HomeAddressStreet>
Home city	<HomeAddressCity>
Home state	<HomeAddressState>
Home postal code	<HomeAddressPostalCode>
Home country	<HomeAddressCountry>
Business address	
Street	<BusinessAddressStreet>
City	<BusinessAddressCity>
State/Prov	<BusinessAddressState>

Postal code	<BusinessAddressPostalCode>
Country	<BusinessAddressCountry>
Other address	
Street	<OtherAddressStreet>
City	<OtherAddressCity>
State/Prov	<OtherAddressState>
Postal code	<OtherAddressPostalcode>
Country	<OtherAddressCountry>
Contact info	
Home phone number	<HomeTelephoneNumber>
Home phone number 2	<Home2TelephoneNumber>
Business phone number1	<BusinessTelephoneNumber>
Business phone number2	<BusinessTelephoneNumber2>
Mobile phone number	<MobileTelephoneNumber>
Secretary phone number	<SecretaryPhone>
Home fax number	<HomeFaxNumber>
Business fax number	<BusinessFaxNumber>
Other mobile phone number	<CarTelephonePhone>
Other phone number	<OtherTelephoneNumber>
Other fax	<OtherFaxNumber>
IstantMessanger	<InstantMessanger>
Emails (1,2,3)	<Email1Address>,<Email2Address>, <Email3Address>[1]
Business info	
Department	<Department>
Profession	<Profession>
Manager	<Manager>
Secretary name	<Secretary>
Additional info	
Business home page	<WebPage>
Text description	<Body>
Subject	<Subject>

6.3. Calendar properties

The following properties are associated to an Exchange calendar item and supported by the connector:

General	
subject	<Subject>

1 In the form “display name” <emailaddress> (e.g.: “John Brown” <john.brown@somewhere.com>)

General	
location	<Location>
dtstart	<Start>
dtend	<End>
duration	<Duration>
alldayevent	<AllDayEvent>
sequence	<Sequence>
timezone	<TimezoneId>
busystatus	<BusyStatus>
meetingstatus	<MeetingStatus>

6.4. Task properties

The following properties are associated to an Exchange task item and supported by the connector:

General	
subject	<Subject>
date	<Date>
description	<Body>

6.5. Note properties

The following properties are associated to an Exchange note item and supported by the connector:

General	
subject	<Subject>
date	<Date>
description	<Body>