



FUNAMBOL mobile open source

Funambol PIM Connector Design Document

Gilberto Migliavacca
Funambol Software Engineer

Funambol
<http://www.funambol.com>

Revision History

<i>Name</i>	<i>Date</i>	<i>Reason for Change</i>	<i>Ver./Rev.</i>
G. Migliavacca	25 September 2006	First draft	1.0
M. Vanetti	20 June 2007	Corrected DB tables	1.1
L. Fassina	18 July 2007	Reviewed section about Database schema	1.2
M. Vanetti	5 October 2007	Added DB table for calendar attendees	1.3
M. Vanetti	23 October 2007	Added another supported value in the DB table for calendar attendees	1.3.1
L. Fassina	30 January 2009	Added section about Twins detection implementation	1.3.2
F. Maggi	18 May 2009	Updated section about Twins detection implementation	1.3.3
F.Machi	07 January 2010	Added section about items not suitable for twin search	1.3.4
L. Fassina	22 December 2010	Changed section about Twins detection implementation	1.4

Table of Contents

1. Introduction.....	5
1.1. Purpose of this Document.....	5
2. Porting the old PIM Components to the new PIM Connector Module.....	6
2.1. Migrating Foundation Module (@todo).....	6
2.2. Migrating PDI Module (@todo).....	6
3. Architecture.....	8
3.1. Main Features.....	8
3.2. Packaging.....	10
3.3. PIM DB feature structure.....	10
3.4. PIM FS feature structure.....	11
4. Classes.....	13
4.1. PIM DB Feature Main Classes.....	13
4.1.1. com.funambol.connector.pim.engine.source.PIMSyncSource and specific SyncSource	13
4.1.2. com.funambol.connector.pim.admin.PIMSyncSourceConfigPanel.....	15
4.1.3. com.funambol.connector.pim.items.manager.PIMEntityManager.....	15
4.1.4. com.funambol.connector.pim.items.dao.PIMEntityDAO.....	16
4.1.5. com.funambol.connector.pim.items.model.***Wrapper.....	17
4.1.6. com.funambol.connector.pim.security.PIMOfficer.....	17
4.1.7. com.funambol.connector.pim.exception.***Exception.....	17
4.1.8. com.funambol.connector.pim.items.model.***Filter.....	17
4.2. PIM FS Feature Main Classes.....	18
4.2.1. com.funambol.foundation.engine.source.AbstractFileSystemSyncSource20	
4.2.2. com.funambol.foundation.engine.source.AbstractSIFSyncSource.....	20
4.2.3. com.funambol.foundation.engine.source.FileSystemSyncSource.....	20
4.2.4. com.funambol.foundation.engine.source.SIFVCardSyncSource.....	20
4.2.5. com.funambol.foundation.engine.source.SIFICalSyncSource.....	20
4.2.6. com.funambol.foundation.engine.source.SIFSyncSource.....	20
5. PIM Configuration Panel.....	21

5.1. PIM Connector Admin Panels (PIM DB feature).....	21
5.2. PIM Connector Admin Panels (PIM FS feature).....	22
6. Twins detection implementation.....	24
7. Filtering PIM Entity.....	30
8. PIMContainer Listener.....	31
8.1. Java Listener Based Architecture.....	31
8.2. DB Trigger Based Architecture.....	32
9. SQL Scripts.....	33
9.1. Funambol PIM Module Registration.....	33
10. Database Schema.....	36
11. Appendices.....	42
11.1. Appendix A – References.....	42

1. Introduction

1.1. Purpose of this Document

The purpose of this document is to describe the technical architecture and design of the Funambol PIM Connector Module

The first chapter describes the old components that will be included in the new Funambol PIM Connector Module.

In the next chapters will be described the new packages and classes that will be created “ad hoc” for the new component.

This document is intended to be read by the development and design team.

2. Porting the old PIM Components to the new PIM Connector Module

2.1. Migrating Foundation Module (@todo)

The current implementation of this module contains some java classes that will be removed and inserted in the new Funambol PIM Connector.

In the following table we list the components that will be included in the PIM Connector

<i>Old package/component</i>	<i>New package/component</i>
modules\foundation\src\java\com\funambol\foundation\admin	<cv\$_prefix>\src\java\com\funambol\pim\admin
modules\foundation\src\java\com\funambol\foundation\engine	<cv\$_prefix>\src\java\com\funambol\pim\engine
modules\foundation\src\java\com\funambol\foundation\phones	<cv\$_prefix>\src\java\com\funambol\pim\phones
modules\foundation\src\java\com\funambol\foundation\synclet	<cv\$_prefix>\src\java\com\funambol\pim\synclet
modules\foundation\src\bean\com\funambol\server\engine\pipeline\bsh	<cv\$_prefix>\src\bean\com\funambol\server\engine\pipeline\bsh
modules\pdi\src\sql	<cv\$_prefix>\src\sql

Note: <cv\$_prefix> = [ObjectWeb-CVS]/sync4j/funambol/modules/pim

the installation procedure inserts the following syncsource

- FileSystem SyncSource

2.2. Migrating PDI Module (@todo)

The current implementation of this module doesn't contains java classes (the source classes are included in the foundation package) but it includes the sql script for the installation of the following SyncSource:

- FileSystem SyncSource

- note
- briefcase
- SIF SyncSource
 - SIF-C
 - SIF-E
 - SIF-T
 - SIF-N
- VCard SyncSource
 - text/x-vcard version 2.1
- ICal SyncSource
 - text/x-vcalendar version 1.0

This package will be removed from the CVS and all the sql scripts will be included in the Funambol PIM Connector

In the following table we list the components that will be included in the PIM Connector

<i>Old package/component</i>	<i>New package/component</i>
modules\pdi\src\bean	<cvb_prefix>\src\bean
modules\pdi\src\db	<cvb_prefix>\src\db
modules\pdi\src\install	<cvb_prefix>\src\install
modules\pdi\src\sql	<cvb_prefix>\src\sql

Note: <cvb_prefix> = [ObjectWeb-CVS]/sync4j/funambol/modules/pim

3. Architecture

3.1. Main Features

This component is an usual Funambol connector and it allows the Funambol DS-Server to read and write PIM entities from and to

- a PIM DB Schema (the PIM DB Schema will be described in the next Chapter)
- a FileSystem structure (this feature is an heritage of the old PDI Module implementation)

Accordingly to the requirements document, the features to implement are summarized in the following list:

- Efficient adapter to an PIM DB Schema
- Efficient adapter to a FileSystem structure
- Supported operations:
 - `getAllItems()` for a given user/principal
 - `getNewItem()` for a given user/principal since a point in time
 - `getUpdateItems()` for a given user/principal since a point in time
 - `getDeletedItems()` for a given user/principal since a point in time
 - `addItem()` for a given user
 - `updateItem()` for a given user
 - `deleteItem()` for a given user
- Easy configuration
- Supported item types:
 - Contacts
 - Events
 - Tasks (todo)
 - Notes ()
- Synclet feature provisioning

For this implemented model each user owns his/her contacts, events, notes and tasks and can see only his/her own data.

In the Figure 1 we show the Funambol PIM Connector main architecture

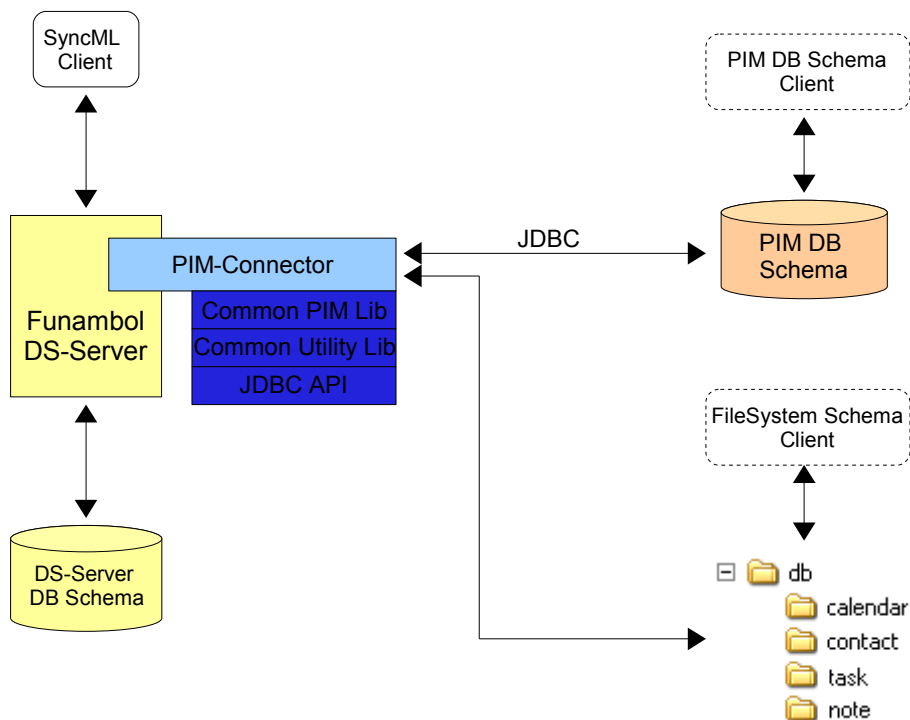


Figure 1: Schema

The Funambol PIM connector will be two main feature:

- adapter for DB called “**PIM DB feature**” in the next chapters: this feature allows the connector to manage the communication with the PIM DB Schema
- adapter for FileSystem called “**PIM FS feature**” in the next chapters: this feature allows the connector to manage the communication with a FileSystem

In the following table we list the packages that will be included in the Funambol PIM Connector

<i>Old package/component (in the foundation and pdi modules)</i>	<i>New package/component</i>	<i>Note</i>
modules\foundation\src\java\com\funambol\foundation\admin	<cv_s_prefix>\src\main\java\com\funambol\pim\admin	
modules\foundation\src\java\com\funambol\foundation\engine	<cv_s_prefix>\src\main\java\com\funambol\pim\engine	
modules\foundation\src\java\com\funambol\foundation\phones	<cv_s_prefix>\src\main\java\com\funambol\pim\phones	
modules\foundation\src\java\com\funambol\foundation\synclet	<cv_s_prefix>\src\main\java\com\funambol\pim\synclet	
modules\foundation\src\bean\com\funambol\server\engine\pipeline\bsh	<cv_s_prefix>\src\main\bean\com\funambol\server\engine\pipeline\bsh	
modules\pdi\src\sql	<cv_s_prefix>\src\main\sql	

Note: <cv_s_prefix> = [ObjectWeb-CVS]/sync4j/funambol/modules/pim

3.2. Packaging

The java classes developed for the Funambol PIM Connector module are packaged under the *com.funambol.connector.pim* package.

Subpackages are created as required, the main of them are:

*.admin	SyncAdmin related classes. It will include the current PDI Configuration panel and the new Configuration panel for the Funambol PIM connector
*.engine.recovery	@todo
*.engine.source	Sync source related classes. It will include the current PDI SyncSource (AbstractFileSystemSyncSource.java, AbstractSIFSyncSource.java, FileSystemSyncSource.java, FilterableSIFSyncSource.java, SIFICalSyncSource.java, SIFSyncSource.java, SIFVCardSyncSource.java) and the new PIM SyncSource (PIMSyncSource, PIMContactSyncSource, PIMCalendarSyncSource, PIMTaskSyncSource, PIMNoteSyncSource)
*.exception	Exception related classes
*.item	Items related classes. (dao, manager and model)
*.item.dao *.item.manager *.item.model	It will contains the classes for the “three layer” structure for the Funambol PIM connector (see 3.3 paragraph)
*.security	Officer related class
*.util	utility classes

The java classes developed for the synclet feature are packaged under the *com.funambol.pim* package.

Subpackages are created as required, the main of them are:

*.synclet	@todo
*.phones	Old implementation of the @todo

3.3. PIM DB feature structure

The PIM DB feature in the Funambol PIM Connector will have a 'three layers' structure as the Figure 2 shows:

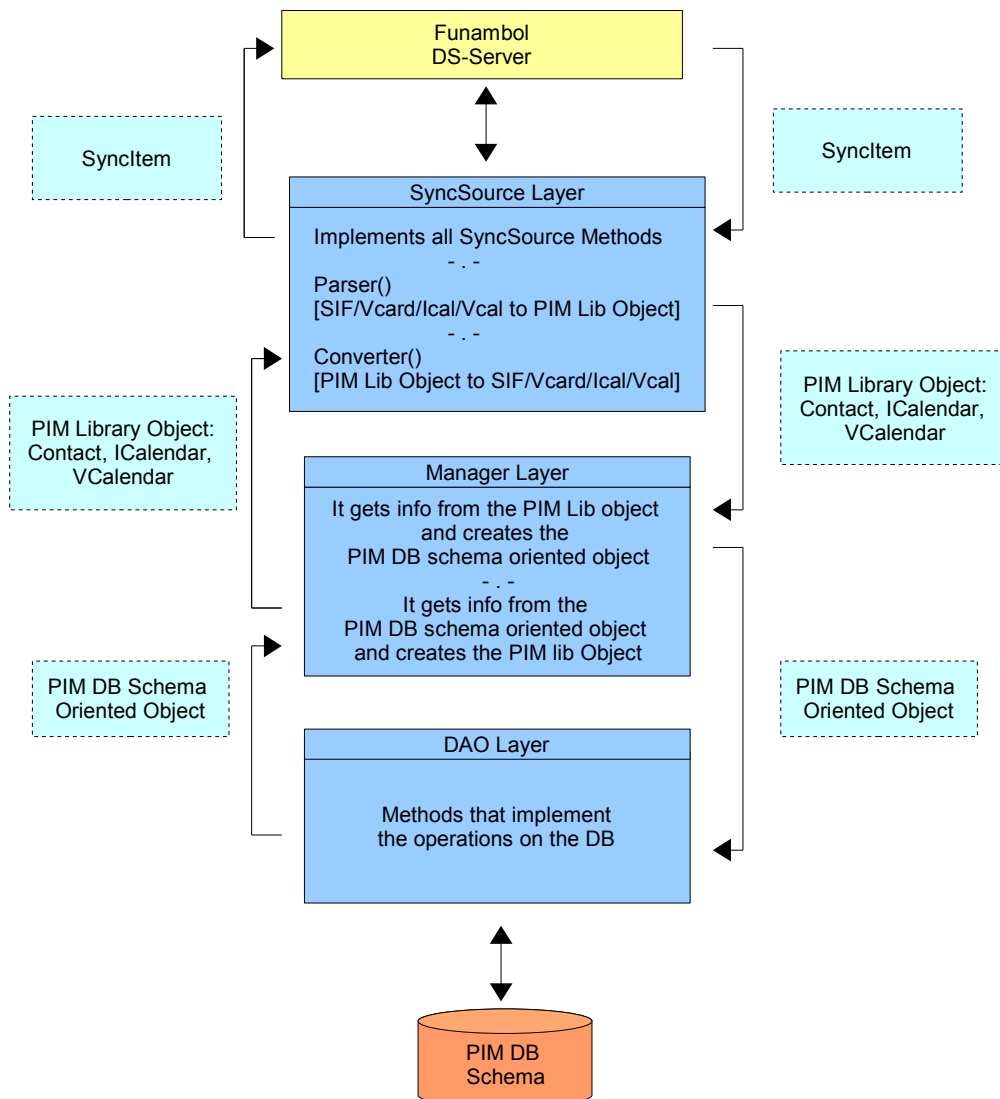


Figure 2: Three Layer structure of the DB adapter feature

- **syncsource layer:** this layer receives the SyncItem from the DS-Server and it implements all the SyncSource Methods defined by the DS-Server for the SyncSource Interface. In this layer will be the calls to the parser (for the “input” direction) and to the converter (for the “output” direction).
- **manager layer:** this layer handles PIM Library Object and the PIM DB Schema Object; it converts the PIM Library Object in to the PIM DB Schema Object (for the “input” direction) and viceversa (for the “output” direction).
- **dao layer:** this layer provides the operations on the PIM DB Schema.

3.4. PIM FS feature structure

The PIM DB feature in the Funambol PIM Connector will have the structure as the Figure 3 shows:

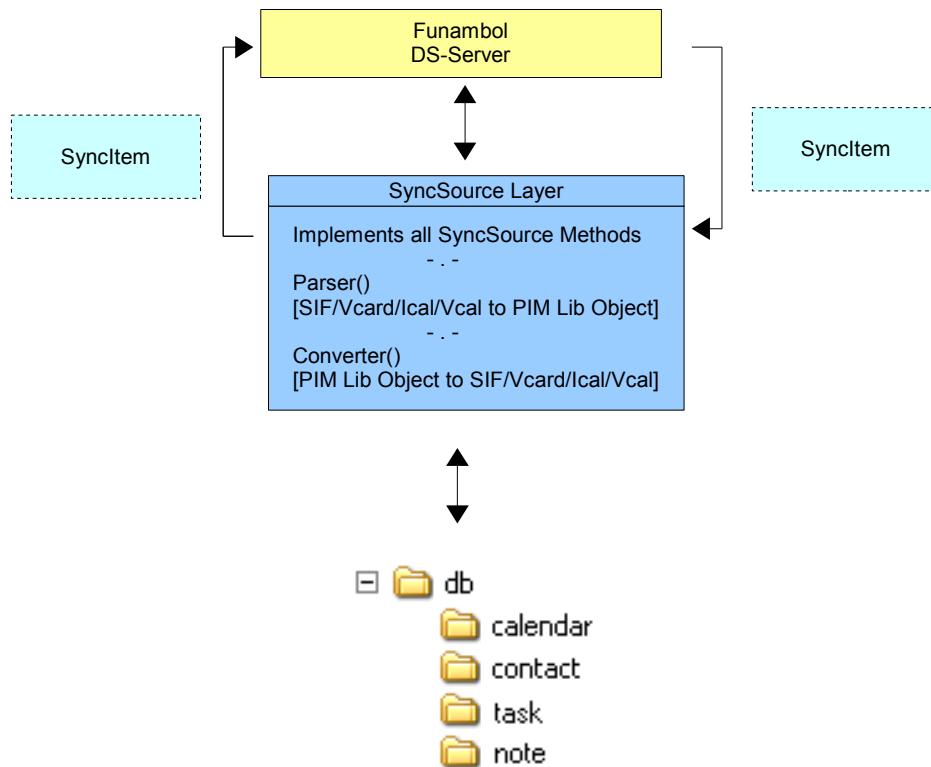


Figure 3: Three Layer structure of the FileSystem adapter feature

- **syncsource layer:** this layer receives the SyncItem from the DS-Server and it implements all the SyncSource Methods defined by the DS-Server for the SyncSource Interface. In this layer will be the calls to the parser (for the “input” direction) and to the converter (for the “output” direction). Moreover all the methods handle the operations on the FileSystem

4. Classes

4.1. PIM DB Feature Main Classes

In this paragraph we analyze the main classes of the new Funambol PIM Connector.

4.1.1. `com.funambol.connector.pim.engine.source.PIMSyncSource` and specific `SyncSource`

The `PIMSyncSource` implements the common method for the specific `PIM***SyncSource`; the list of the specific syncsource classes is:

- `PIMContactSyncSource`: it should handle the Contact if the client sends the item as:
 - SIF-C format
 - vcard format
 - `PIMSIFCalendarSyncSource`: it should handle the Event if the client send the item in:
 - SIF-E format.
 - `PIMSIFTaskSyncSource`: it should handle the Task if the client sends the item in:
 - SIF-T format
 - `PIMWebCalendarSyncSource`: it should handle the Event and the Task if the client send the item in:
 - icalendar/vcalendar format.
- Note It may happen that some mobile phone sends in the same session Event and Todo entity
- `PIMNoteSyncSource`: it should handle the Note if the client sends the item in:
 - SIF-N format
 - vnote format

The Figure 4 shows the diagram class of the `PIMSyncSource` and `PIMContactSyncSource` (it's the same for the rest of the specific `SyncSource`)

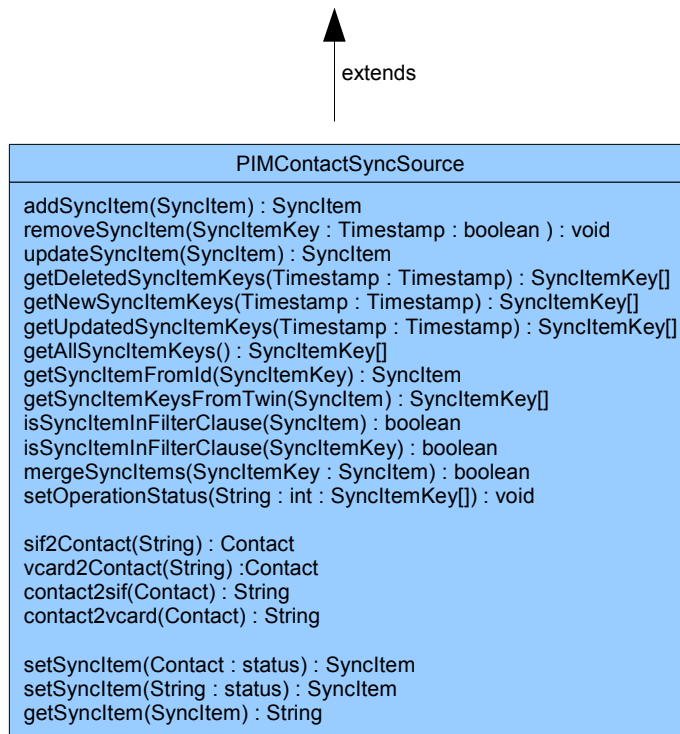


Figure 4: *SyncSource Diagram Class**

4.1.2. com.funambol.connector.pim.admin.PIMSyncSourceConfigPanel

This class allows the user to set the properties for the PIMSyncSource (for detail see Chapter 5)

4.1.3. com.funambol.connector.pim.items.manager.PIMEntityManager

The PIMEntityManager implements the common method for the specific PIM***EntityManager; the list of the specific manager classes is:

- PIMContactEntityManager
- PIMClandarEntityManager
- PIMTaskEntityManager
- PIMNoteEntityManager

The Figure 5 shows the diagram class of the PIMEntityManager and PIMContactEntityManager

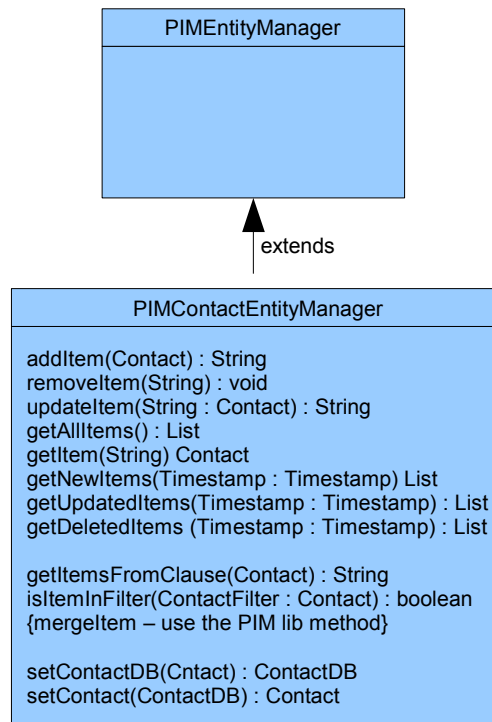


Figure 5: *EntityManager Diagram Class**

4.1.4. com.funambol.connector.pim.items.dao.PIMEntityDAO

In this class will be all the method to operate on the DB.

The PIMEntityDAO implements the common method for the specific PIM***EntityDAO; the list of the specific DAO classes is:

- PIMContactEntityDAO
- PIMClandarEntityDAO
- PIMTaskEntityDAO
- PIMNoteEntityDAO

The Figure 6 shows the diagram class of the PIMEntityDAO and PIMContactEntityDAO

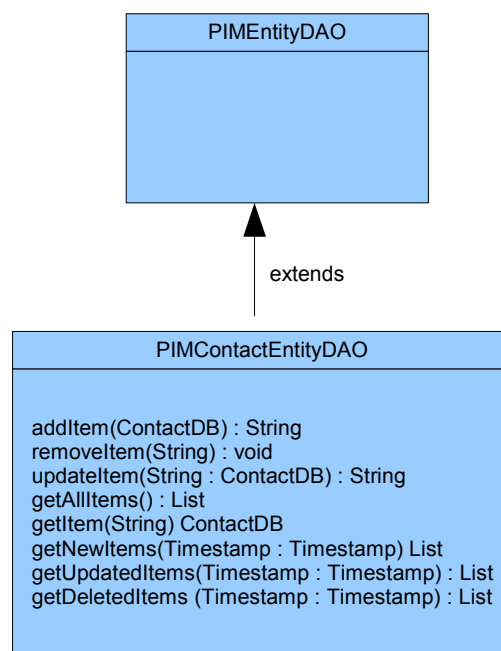


Figure 6: *EntityDAO Diagram Class**

We describe how some methods should work in the DAO layer **@tocheck** (for detail about the fields used in the following description see the chapter 8 "PIM DB Schema")

getNewItem (Timestamp since : Timestamp to) : List

This method should return the list of the items that satisfy the rule:

- status = "N" and "since" < last_modification

getUpdatedItems (Timestamp since : Timestamp to) : List

This method should return the list of the items that satisfy the rule:

- status = "U" and "since" < last_modification

getDeletedItems (Timestamp since : Timestamp to) : List

This method should return the list of the items that satisfy the rule:

- status = "D" and "since" < last_modification

4.1.5. com.funambol.connector.pim.items.model.***Wrapper

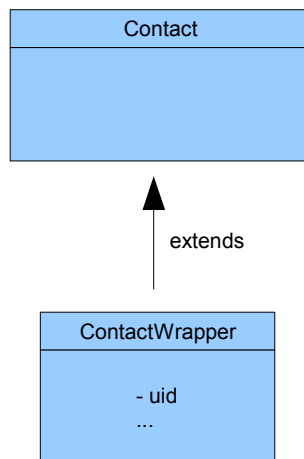
This classes is the model for the fields in the database.

for instance the ContactWrapper extends com.funambol.common.pim.contact.Contact and adds some attributes like:

- uid
- @tocheck

There will be the following classes:

- ContactWrapper.java
- CalendarWrapper.java
- TaskWrapper.java
- NoteWrapper.java



4.1.6. com.funambol.connector.pim.security.PIMOfficer

This class should implement the authentication procedure.

4.1.7. com.funambol.connector.pim.exception.***Exception

We define two type of Exception

- com.funambol.pim.exception.EntityException: This Exception should be used when a statement throws an error in the PIM entity handling steps
- com.funambol.pim.exception.PIMDBAccessException: This Exception should be used when a statement throws an error during the connection/disconnection to the DB

4.1.8. com.funambol.connector.pim.items.model.***Filter

This classes will be used for save the filter clauses that the client sends.

There will be the following classes:

- ContactFilter.java
- CalendarFilter.java
- TaskFilter.java
- NoteFilter.java

In the beginSync() method will be initialized for the entire sync session the specific filter class.

4.2. PIM FS Feature Main Classes

In this paragraph we analyze the main classes that will be migrated from the current “funambol Foundation Module” (see Chapter 1).

The Figure 7 shows the digram class for the PIM FS feature main classes.

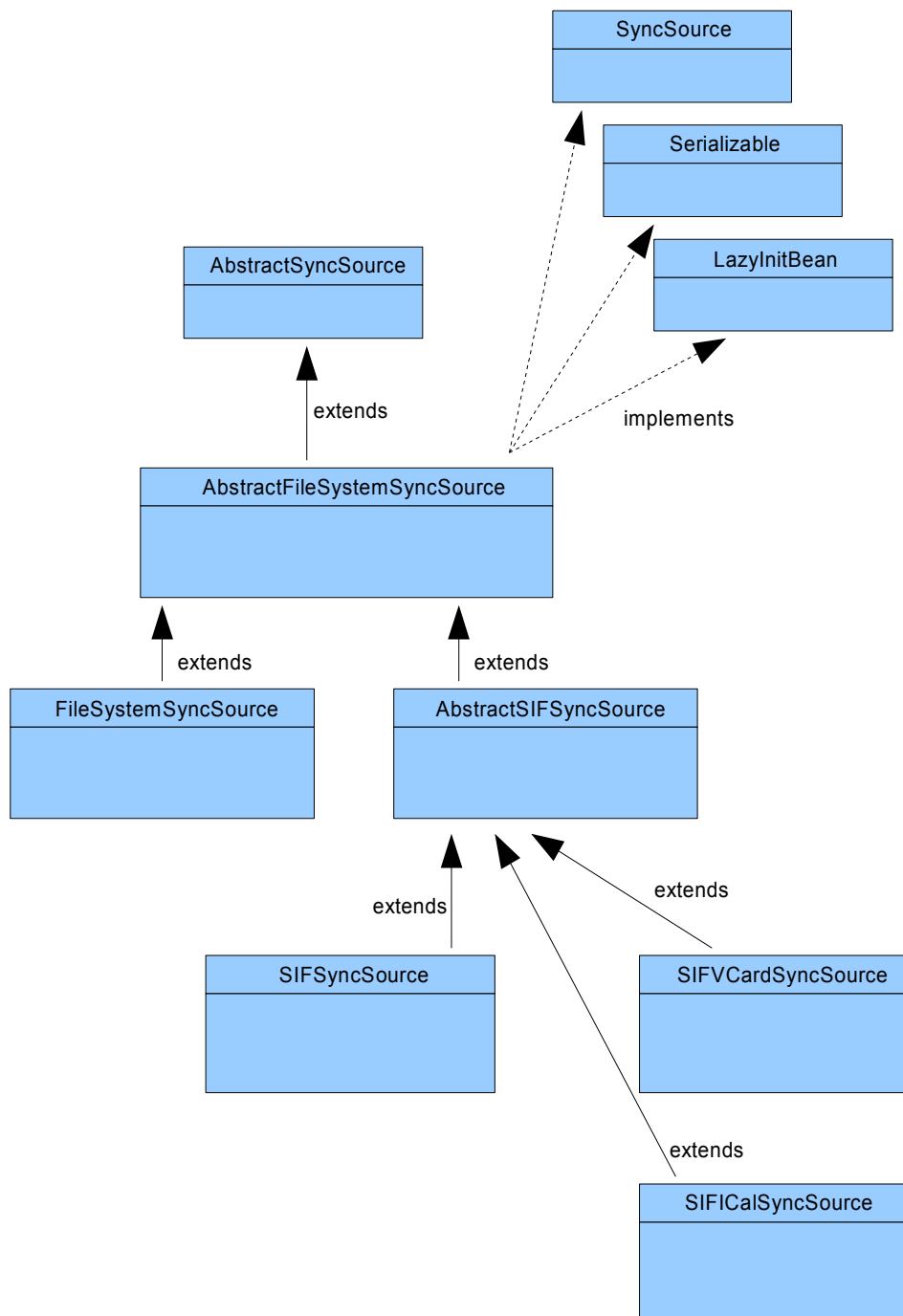


Figure 7: SyncSource Diagram Class

4.2.1. com.funambol.foundation.engine.source.AbstractFileSystemSyncSource

It contains common methods for all the SyncSource in this paragraph. It define also the abstract methods that should be implemented in the specific SyncSource.

Note: we don't describe the diagram class because the class is already implemented (see Funambol Fondation Module)

4.2.2. com.funambol.foundation.engine.source.AbstractSIFSyncSource

It contains common methods for the SIF***SyncSource.

Note: we don't describe the diagram class because the class is already implemented (see Funambol Fondation Module)

4.2.3. com.funambol.foundation.engine.source.FileSystemSyncSource

This class handles generic items and save/get/remove them on the FileSystem

Note: we don't describe the diagram class because the class is already implemented (see Funambol Fondation Module)

4.2.4. com.funambol.foundation.engine.source.SIFVCardSyncSource

This class handles items in the VCARD format (text/x-vcard) and save/get/remove them on the FileSystem

Note: we don't describe the diagram class because the class is already implemented (see Funambol Fondation Module)

4.2.5. com.funambol.foundation.engine.source.SIFICalSyncSource

This class handles items in the ICAL format (text/x-vcalendar) and save/get/remove them on the FileSystem

Note: we don't describe the diagram class because the class is already implemented (see Funambol Fondation Module)

4.2.6. com.funambol.foundation.engine.source.SIFSyncSource

This class handles items in the SIF-C/SIF-E/SIF-T/SIF-N formats and save/get/remove them on the FileSystem

Note: we don't describe the diagram class because the class is already implemented (see Funambol Fondation Module)

5. PIM Configuration Panel

5.1. PIM Connector Admin Panels (PIM DB feature)

The Figure 8 shows the Configuration panel for the Calendar SyncSource

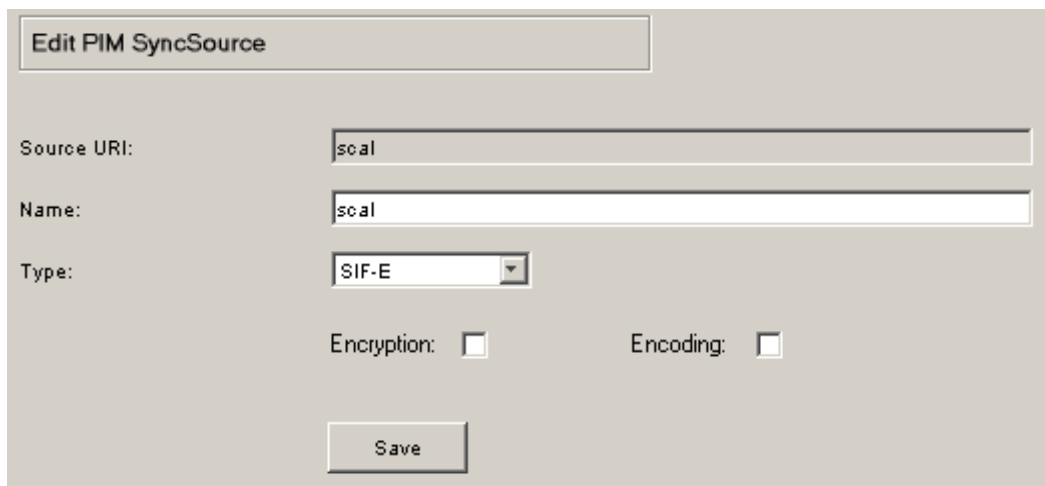


Figure 8: SyncSource Configuration Panel

this is the list of the Contact SyncSource parameters:

<i>Property</i>	<i>Description</i>
Source URI	The sync source URI
Name	The SyncSource name
Supported Types	The combo-box shows: SIF-C and VCARD
Encryption/Encoding	Should the Data content be encrypted using DES algorithm? Should the Data content be encoded using Base64 algorithm?

this is the list of the SCalendar SyncSource parameters:

<i>Property</i>	<i>Description</i>
Source URI	The sync source URI
Name	The SyncSource name
Supported Types	Just SIF-E
Encryption/Encoding	Should the Data content be encrypted using DES algorithm? Should the Data content be encoded using Base64 algorithm?

this is the list of the STask SyncSource parameters:

<i>Property</i>	<i>Description</i>
Source URI	The sync source URI
Name	The SyncSource name
Supported Types	Just SIF-T
Encryption/Encoding	Should the Data content be encrypted using DES algorithm? Should the Data content be encoded using Base64 algorithm?

this is the list of the ICalendar SyncSource parameters:

<i>Property</i>	<i>Description</i>
Source URI	The sync source URI
Name	The SyncSource name
Supported Types	Just ICAL (it includes VEVENT and VTODO)
Encryption/Encoding	Should the Data content be encrypted using DES algorithm? Should the Data content be encoded using Base64 algorithm?

this is the list of the Note SyncSource parameters:

<i>Property</i>	<i>Description</i>
Source URI	The sync source URI
Name	The SyncSource name
Supported Types	The combo-box shows: SIF-N and VNOTE
Encryption/Encoding	Should the Data content be encrypted using DES algorithm? Should the Data content be encoded using Base64 algorithm?

5.2. PIM Connector Admin Panels (PIM FS feature)

The Figure 9 shows the SIF SyncSource configuration Panel

Edit SIF SyncSource

Source URI:

Name:

Type:
SIF-C

Source Directory:

Twin properties:
(Used to identify twins)

Anniversary
AssistantName
AssistantTelephoneNumber
Birthday

MultiUser:
☐

Add

Figure 9: SIF SyncSource Configuration Panel

this is the list of the SIF SyncSource parameters:

<i>Property</i>	<i>Description</i>
Source URI	The sync source URI
Name	The SyncSource name
Type	The combo-box shows: SIF-C, SIF-E, SIF-T, SIF-N
Source Directory	Location (directory path) in the FileSystem
Twin Properties	Used to identify the twins (the properties list depends on the selected type)
MultiUser	Enable different directory for different user

6. Twins detection implementation

This chapter explains how the twins detection has been implemented.

The way to find twins is changed more from the previous Funambol versions but only for the contact items.

The SyncSource compares some client item's properties with some server item's properties in order to detect a twin item. These item's properties are called twin fields and every SyncSource can apply any rules to find twin items. If twin fields have the same value both in client item and server item, then these items are considered as twins.

Since the twin search is based on a set of properties, if an item misses all those properties, the server is not able to recognize whether that item has twins or not and, in that case, the server has no means to avoid duplication and the number of duplicated items will grow at each slow sync deteriorating server performances.

In order to handle such scenario in a proper way, the foundation SyncSources reject items not suitable for twin search: if an item has no twin fields set to a meaningful value, any adding of such an item will cause an error. In particular, if a client tries to add a contact (or an event or a task or a note) without any twin field, the status returned by the server for such an item will contain a 506 error code and the corresponding error message:

"Adding a contact without any field usable for twin search set is not allowed."

Besides, since the adding of item without any twin field has been forbidden, the twin search in each SyncSource has been updated in order to skip the search and return an empty list when handling an item with no twin field.

Below, you can find a list of twin search examples.

1. To find a twin of an event, the SyncSource compares *subject*, *start date* and *end date*.

For instance:

Event A	Event B
subject: Release planning start date: 01-02-2009 10:00:00 end date: 01-02-2009 11:00:00 description: meeting	subject: Release planning start date: 01-02-2009 10:00:00 end date: 01-02-2009 11:00:00 description: business meeting

Event B is twin of Event A even if the *description* field is different since this is not a twin field.

Event C	Event D
subject: Release planning start date: 01-02-2009 10:00:00 end date: 01-02-2009 11:00:00 description: meeting	subject: Release planning start date: 01-02-2009 10:00:00 end date: description: meeting

Event D is not twin of Event C since in the Event D the *end date* field is different (it is empty).

2. To find a twin of a task, the SyncSource compares *subject* and *due date*.

For instance:

Task A	Task B
subject: Wash car due date: 04-02-2009 11:00:00	subject: Wash car due date: 04-02-2009 11:00:00 status: complete

Task B is twin of Task A even if it has the *status* field set since this is not a twin field.

Task C	Task D
subject: Wash car due date: 04-02-2009 11:00:00 status: complete	subject: Wash car status: complete

Task D is not twin of Task C since the *due date* is not set.

3. The twin search of the contacts is more complicated than the logic implemented in the other PIM data SyncSources.
It is based on different level of skimming whose result is, first of all, a list of potential twins and then, if exists, the twin contact.

The contacts that satisfy the following rules, will be included in the list of potential twins:

1. first names and last names match OR
2. first names match when there are no last names OR
3. last names match when there are no first names OR
4. there are no first names and no last names, but the display names match OR
5. there are no first names, no last names and no display names, but the company names match

After creating the macro list of potential twins, the following rules will be applied on these contacts:

1. a contact contains no fields other than name (first/last/display/company) and the other contains name plus other field OR
2. the contacts contain at least one identical email address in any of the address fields OR
3. the contacts contain at least one identical phone number in any of the phone number fields

At this point, if the list is still filled, it's needed to take in account also these rules:

1. there are no more than 3 email addresses in total when the contacts are merged in order to prevent data loss (note that this is for future enhancement since now we are able to handle just 3 email address types) AND
2. the contacts don't have different phone numbers and email addresses in the same corresponding field (e.g. 2 different phone numbers in the same HOME phone # fields of the contacts)

For instance:

Contact 1

First name: Lucy
Last name: Parker
Email home: lucy@gmail.com
Phone home: +555 123456

Contact 2

First name: Lucy
Last name: Parker
Email home: lucy@gmail.com
Phone home: +555 987654

Contact 2 IS NOT twin of Contact 1 since the *home phone number* is different.

Contact 3

First name: Lucy
Last name: Parker
Email home:
Phone home: +555 123456

Contact 4

First name: Lucy
Last name: Parker
Email home: lucy@gmail.com
Phone home: +555 123456

Contact 4 IS twin of Contact 3 even if the *home email address* is not set.

Contact 5

First name:
Last name:
Company name: ElectronicX
Email home: lucy@gmail.com
Phone home: +555 123456

Contact 6

First name:
Last name:
Company name: Rent car
Email home: lucy@gmail.com
Phone home: +555 123456

Contact 6 IS NOT twin of Contact 5 since the *company name* is different.

Contact 7

First name:
Last name:
Company name: ElectronicX
Email home: parker.lucy@gmail.com
Phone home: +555 123456

Contact 8

First name:
Last name:
Company name: ElectronicX
Email work: lucy@gmail.com
Phone home: +555 123456

Contact 8 IS twin of Contact 7 since the *email address* are different but their types are different too and the number of email after merging is less than 3.

Contact 9

First name:
Last name:
Company name:
Display name:
Email work:
Email home: lp@gmail.com
Phone home: +555 92345776

Contact 10

First name:
Last name:
Company name:
Display name:
Email home: lp@gmail.com

Contact 10 IS twin of Contact 9.

Contact 11

First name:
Last name:
Company name:
Display name:
Email home: lp@gmail.com
Phone home: +555 123456

Contact 12

First name:
Last name:
Company name:
Display name:
Email home: lparker@gmail.com
Phone home: +555 123456

Contact 12 IS NOT twin of Contact 11 since the *home email address* is different.

Contact 13	Contact 14
First name: Last name: Company name: Display name: Email work: lp@gmail.com Phone home: +555 123456	First name: Last name: Company name: Display name: Email home: lucy@gmail.com Email work: lp@gmail.com Phone work: +555 123456

Contact 14 IS twin of Contact 13 since they have the same *work email work* have not different *phone number* in the same fields.

Contact 14	Contact 15
First name: Last name: Company name: Display name: Email home: lucy@gmail.com Email work: lp@gmail.com Phone home: +555 123456	First name: Last name: Company name: Display name: Email home: lucy@gmail.com Email work: lp@gmail.com Phone home: +555 327827

Contact 15 IS NOT twin of Contact 14 since the *home phone number* is different.

Contact 16	Contact 17
First name: Lucy Last name: Parker Company name: Display name: Lucy Parker Phone home: +555 92345776	First name: Lucy Last name: Parker Company name: Display name: L Parker Phone work: +555 885533

Contact 17 IS twin of Contact 16, because *first name* and *last name* are set, therefore *display name* is not used in twin search.

Contact 18	Contact 19
First name: Last name: Company name: ElectronicX Display name: Lucy Parker Phone home: +555 92345776	First name: Last name: Company name: Funambol Display name: Lucy Parker Phone home: +555 92345776

Contact 19 IS NOT twin of Contact 18 since the *company name* are different (since the *company name* is set, the *display name* is not used in twin search).

Contact 20

First name: Jane
Last name: Doe
Phone home: +555 92345776
Phone work: +555 12121212

Contact 21

First name: Jane
Last name: Doe
Phone home: +555 12121212
Phone work: +555 92345776

Contact 21 IS NOT twin of Contact 20 since the *phone numbers*, for the same type field, are different.

Contact 22

First name: Jane
Phone work: +555 12121212

Contact 23

First name: Jane
Last name: Doe
Phone work: +555 12121212

Contact 23 IS NOT twin of Contact 22 since the *last name* is different.

Contact 24

First name: Jane
Last name: Doe
Phone work: +555 12121212
Email company: info@company.com

Contact 25

First name: Jane
Phone work: +555 12121212

Contact 25 IS NOT twin of Contact 24 since the last name is different.

Contact 26

First name: Jane
Last name: Doe
Email home: info@home.com

Contact 27

First name: Jane
Last name: Doe
Email work: work@fun.com
Email other: other@fun.com

Contact 27 IS NOT twin of Contact 26 since there is no at least an *email address* in common.

4. To find a twin of a note, the SyncSource compares the *body*.

For instance:

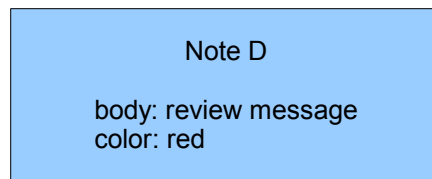
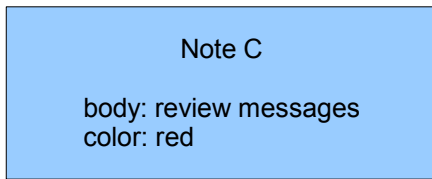
Note A

body: review messages

Note B

body: review messages
color: red

Note A is twin of Note B even if it has not set the *color* property since this is not a twin field.



Note D is not twin of Note C since the *body* is different.

7. Filtering PIM Entity

@todo

8. PIMContainer Listener

This component monitors the: `fnbl_pim_contact`, `fnbl_pim_calendar` tables for set of accounts. This two tables system will be called "*PIMSchema*".

When an update/delete/insert operation is detected in the user "*PIMSchema*", the Funambol's AdminManager is invoked in order to send to the DS-Server alerted sync notification to the user's device.

Since the number of monitored "*PIMSchema*" can be very big, the architecture of this components allows to partition the user base so that many new listeners can be started (maybe on different machines). Each listeners will monitor a subset of the user base "*PIMSchema*".

To implement this schema we can think about two different architecture:

- java listener based architecture
- db trigger based architecture

8.1. Java Listener Based Architecture

The architecture of the PIMContainer Listener component is illustrated in Figure 10.

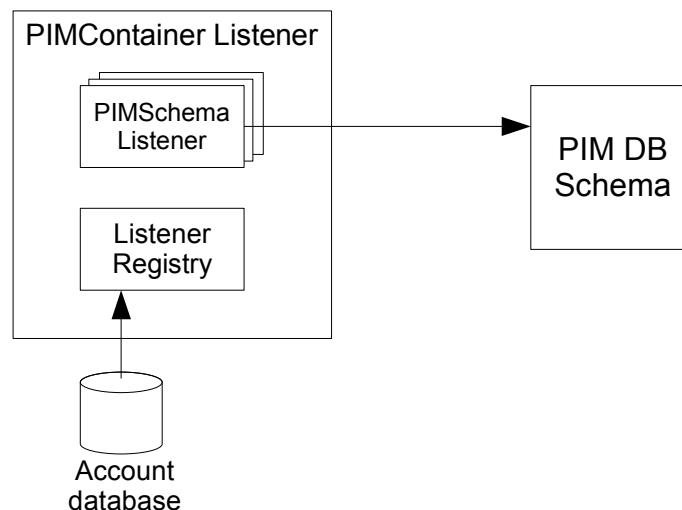


Figure 10: PIMContainer Listener architecture

PIMContainer Listener is a main program that can be started on the command line.

It first starts the ListenerRegister thread which reads the account database and, for each account associated to the running PIMContainer Listener, it registers a new PIM-Schema Listener for the account.

After its first run, ListenerRegistry sleeps for a while and then it will be executed again. At each new execution the registry compares the list of the already registered accounts with the content of the database in order to register new accounts or deregister accounts that do not need to be monitored anymore.

PIMSchema Listener is fundamentally a module that implements a polling system: every 'x' seconds open the connection to the PIM DB Schema and check if there are performed update/delete/insert operation, then it closes the connection.

When a update/delete/insert operation appears in a monitored PIM DB Schema, the corresponding listener calls the Funambol AdminManager web service in order to trigger the push message. This call can be done asynchronously based on how the PIMContainer Listener is configured.

Note: how to check if there are “performed update/delete/insert operation” @todo

8.2. DB Trigger Based Architecture

The architecture of the PIM-Container Listener component is illustrated in Figure 11.

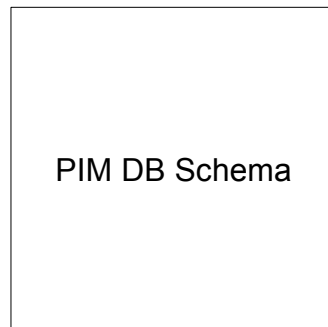


Figure 11: PIMContainer Listener architecture

@todo

9. SQL Scripts

In this chapter we describe the script for the Funambol PIM Module.

9.1. Funambol PIM Module Registration

The `init_schema.sql` must contain the SQL code that registers each piece of the connector. The goal is to see the hierarchy of Figure 12

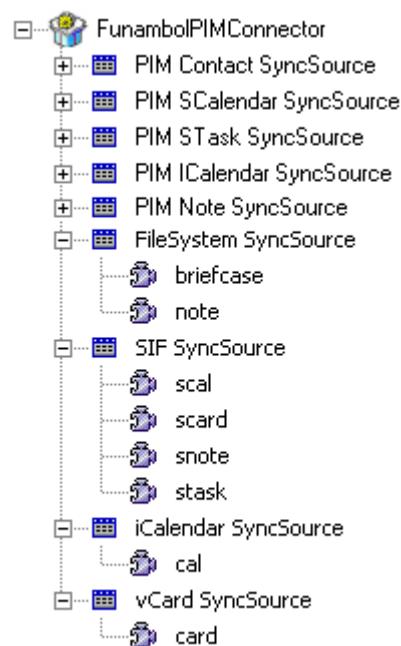


Figure 12: Hierarchy of the Funambol PIM Connector

To achieve this, the following data must be inserted:

FNBL_MODULE		
moduleid	modulename	description
pim	pim	Funambol PIM Connector

FNBL_CONNECTOR			
<i>id</i>	<i>name</i>	<i>description</i>	<i>admin_class</i>
pim	FunambolPIMConnector	Funambol PIM Connector	

FNBL_SYNC_SOURCE_TYPE			
<i>id</i>	<i>description</i>	<i>class</i>	<i>admin_class</i>
pim-co	PIM Contact SyncSource	com.funambol.connector.pim.engine.s ource.PIMContactSyncSource	com.funambol.connector.pim.admin .PIMSyncSourceConfigPanel
pim-sca	PIM SIF Calendar SyncSource	com.funambol.connector.pim.engine.s ource.PIMSCalendarSyncSource	com.funambol.connector.pim.admin .PIMSyncSourceConfigPanel
pim-sta	PIM SIF Task SyncSource	com.funambol.connector.pim.engine.s ource.PIMSTaskSyncSource	com.funambol.connector.pim.admin .PIMSyncSourceConfigPanel
pim-ical	PIM ICal SyncSource	com.funambol.connector.pim.engine.s ource.PIMICalSyncSource	com.funambol.connector.pim.admin .PIMSyncSourceConfigPanel
pim-no	PIM Note SyncSource	com.funambol.connector.pim.engine.s ource.PIMNoteSyncSource	com.funambol.connector.pim.admin .PIMSyncSourceConfigPanel
fs-pdi	FileSystem SyncSource	com.funambol.connector.pim.engine.s ource.FileSystemSyncSource	com.funambol.connector.pim.admin .FileSystemSyncSourceConfigPane l
sifvc-pdi	vCard SyncSource	com.funambol.connector.pim.engine.s ource.SIFVCardSyncSource	com.funambol.connector.pim.admin .SIFVCardSyncSourceConfigPanel
sific-pdi	iCalendar SyncSource	com.funambol.connector.pim.engine.s ource.SIFICalSyncSource	com.funambol.foundation.admin.SI FICalSyncSourceConfigPanel
sif-pdi	SIF SyncSource	com.funambol.connector.pim.engine.s ource.SIFSyncSource	com.funambol.connector.pim.admin .SIFSyncSourceConfigPanel

FNBL_MODULE_CONNECTOR	
<i>module</i>	<i>connector</i>
pim	pim

FNBL_CONNECTOR_SOURCE_TYPE	
<i>connector</i>	<i>sourcetype</i>
pim	pim-co
pim	pim-sca
pim	pim-sta
pim	pim-ical
pim	pim-no
pim	fs-pdi
pim	sifvc-pdi
pim	sific-pdi
pim	sif-pdi

The next table describe the values in order to create the FileSystem Adapter feature SyncSource

FNBL_SYNC_SOURCE			
<i>uri</i>	<i>config</i>	<i>name</i>	<i>sourcetype</i>
briefcase	pim/pim/fs-pdi/BriefcaseSource.xml	briefcase	fs-pdi
note	pim/pim/fs-pdi/NoteSource.xml	note	fs-pdi
card	pim/pim/sifvc-pdi/ContactSource.xml	card	sifvc-pdi
cal	pim/pim/sific-pdi/CalendarSource.xml	cal	sific-pdi
scal	pim/pim/sif-pdi/SIFCalendarSource.xml	scal	sif-pdi

<i>FNBL_SYNC_SOURCE</i>			
scard	pim/pim/sif-pdi/SIFContactSource.xml	scard	sif-pdi
stask	pdi/pdi/sif-pdi/SIFTaskSource.xml	stask	sif-pdi
snote	pdi/pdi/sif-pdi/SIFNoteSource.xml	snote	sif-pdi

10. Database Schema

The installation procedure of the Funambol PIM Connector will provide the creation of the PIM DB Schema.

In tables below we list the fields for the tables in the PIM DB Schema

fnbl_pim_contact contains the information about the contact.

<i>fnbl_pim_contact</i>			
<i>Column</i>	<i>Type</i>	<i>Constraints</i>	<i>Description</i>
id	bigint	PK	Contact id
userid	varchar(255)		User id
last_update	bigint		Last updating time
status	char		Contact status N – New U – Updated D – Deleted
importance	smallint		Priority
sensitivity	smallint		The type of access class
subject	varchar(255)		Subject
folder	varchar(255)		The folder in which save the contact
anniversary	varchar(16)		Anniversary date
first_name	varchar(64)		First name
middle_name	varchar(64)		Middle name
last_name	varchar(64)		Last name
display_name	varchar(128)		Display name
birthday	varchar(16)		Birthday date
body	varchar(255)		Note
categories	varchar(255)		Categories the contact belongs to
children	varchar(255)		Name of the contact's children
hobbies	varchar(255)		List of hobbies
initials	varchar(16)		Initials
languages	varchar(255)		List of languages spoken by contact
nickname	varchar(64)		Nickname

<i>fnbl_pim_contact</i>			
spouse	varchar(128)		Full name of the contact's spouse
suffix	varchar(32)		Suffix name
title	varchar(32)		Salutation (word that preceedes the full name)
gender	char(1)		Gender
assistant	varchar(128)		Full name of the contact's assistant
company	varchar(255)		Company name in which the contact works
department	varchar(255)		Department name in which the contact works
job_title	varchar(128)		Job title
manager	varchar(128)		Full name of the contact's manager
mileage	varchar(16)		Mileage
office_location	varchar(64)		Location of the contact's office
profession	varchar(64)		Professional role of the contact
companies	varchar(255)		Companies the contact is related to

fnbl_pim_contact_item contains the information about phone numbers, emails, web pages and address labels.

<i>fnbl_pim_contact_item</i>			
<i>Column</i>	<i>Type</i>	<i>Constraints</i>	<i>Description</i>
contact	bigint	PK, FK(fnbl_pim_contact)	Contact id
type	smallint	PK	Item type
value	varchar(255)		Item value

fnbl_pim_address contains the information about address home, business and other.

<i>fnbl_pim_address</i>			
<i>Column</i>	<i>Type</i>	<i>Constraints</i>	<i>Description</i>
contact	bigint	PK, FK(fnbl_pim_contact)	Contact id
type	smallint	PK	Address type 1 – Home 2 – Business 3 – Other
street	varchar(128)		Street
city	varchar(64)		City
state	varchar(64)		State
postal_code	varchar(16)		Postal code
country	varchar(32)		Country

<i>fnbl_pim_address</i>			
po_box	varchar(16)		Post office box

fnbl_pim_calendar contains information about event and task calendar.

<i>fnbl_pim_calendar</i>			
<i>Column</i>	<i>Type</i>	<i>Constraints</i>	<i>Description</i>
id	bigint	PK	Calendar id
userid	varchar(255)		User id
last_update	bigint		Last updating time
status	char		Calendar status N – New U – Updated D – Deleted
type	smallint		Calendar's type 1 – Event 2 – Task
all_day	char(1)		Is the calendar an all-day?
body	varchar(255)		Detailed description
busy_status	smallint		Availability of the user during the calendar
categories	varchar(255)		Categories the calendar belongs to
companies	varchar(255)		Companies related to the calendar
birthday	varchar(16)		????????
duration	integer		Calendar's duration
dstart	timestamp		Starting date of the calendar
dend	timestamp		Ending date of the calendar
folder	varchar(255)		The folder in which save the calendar
importance	smallint		Priority level
location	varchar(255)		Location
meeting_status	smallint		Point reached by the calendar's organization process 0 – no meeting 1 – meeting 3 – received 5 – canceled
mileage	varchar(16)		Mileage
reminder_time	timestamp		Date and Time of when the calendar's reminder has to be triggered
reminder	char(1)		Is there a reminder for the

<i>fnbl_pim_calendar</i>			
			calendar ?
reminder_sound_file	varchar(255)		Sound file to be played when the calendar's reminder gets active
reminder_options	integer		Optional features
reminder_repeat_count	integer		Number of times the reminder action has to be repeated
sensitivity	smallint		The type of access class
subject	varchar(1000)		Subject
rec_type	smallint		Type of recurrence rule
rec_interval	integer		Recurrence interval
rec_month_of_year	smallint		Month of the year when the calendar has to be repeated
rec_day_of_month	smallint		Day of month when the calendar has to be repeated
rec_day_of_week_mask	varchar(16)		Day(s) of the week when the calendar has to be repeated
rec_instance	smallint		Instance of the days prescribed by the recurrence period and by the modifiers
rec_start_date_pattern	varchar(32)		Starting date of the recurrence
rec_no_end_date	char(1)		Does the recurrence have the end date?
rec_end_date_pattern	varchar(32)		Ending date of the recurrence
rec_occurrences	smallint		Number of time for which repeat the calendar
reply_time	timestamp		Date and Time when the recipient replied to the meeting request associated with the calendar
completed	timestamp		Date and Time when the Task has been completed
percent_complete	smallint		Task's completion percentage

fnbl_pim_calendar_exception contains information about the calendar exceptions.

<i>fnbl_pim_calendar_exception</i>			
<i>Column</i>	<i>Type</i>	<i>Constraints</i>	<i>Description</i>
calendar	bigint	PK, FK(fnbl_pim_calendar)	Calendar id
addition	char(1)	PK	Whether the exception is an addition or a subtraction
occurrence_date	timestamp	PK	Date and time of the exception

fnbl_pim_calendar_attendee contains information about calendar attendees.

<i>fnbl_pim_calendar_attendee</i>			
<i>Column</i>	<i>Type</i>	<i>Constraints</i>	<i>Description</i>
calendar	bigint	PK, FK(fnbl_pim_calendar)	Calendar id
name	char(128)	PK	Displayed name of the attendee
uri	char(255)	PK	URI related with the attendee (often, a MAILTO link)
role	smallint		Role of the “attendee” in the event: -1 – unknown 0 – true attendee 1 – delegate 2 – organizer 3 – owner
expected	smallint		Presence expectation on the attendee: -1 – unspecified 0 – no participation 1 – optional participation 2 – required participation 3 – required participation with immediate response needed 4 – required participation as a chairman
kind	smallint		Type of attendee: -1 – unknown 0 – individual 1 – group 2 – resource 3 – room
status	smallint		Actual participation status currently declared by the attendee: -1 – unspecified 0 – declined 1 – request needs action 2 – request sent 3 – delegated to somebody else 4 – tentatively accepted 5 – accepted 6 – in the process of completing the subtask assigned to the attendee 7 – subtask assigned to the

<i>fnbl_pim_calendar_attendee</i>			
			attendee completed

fnbl_push_listener_registry contains the information about the task to execute from the framework push listener.

<i>fnbl_push_listener_registry</i>			
<i>Column</i>	<i>Type</i>	<i>Constraints</i>	<i>Description</i>
id	bigint	PK	Push listener registry id
id_push_listener	bigint		Push listener id
period	bigint		Refresh Interval of the information on the source
active	char(1)		Is the task activated?
last_update	bigint		Last time task's execution
status	varchar(1)		Task's status
task_bean_file	varchar(255)		File xml of which the task is the instance

fnbl_pim_listener_registry contains the options of the push for a specified user.

<i>fnbl_pim_listener_registry</i>			
<i>Column</i>	<i>Type</i>	<i>Constraints</i>	<i>Description</i>
id	bigint	FK(fnbl_push_listener_registry)	Push id
username	varchar(50)		User id
push_contacts	char(1)		Is the push for contacts activated?
push_calendars	char(1)		Is the push for calendars activated?

The **last_update** and **status** fields in the *fnbl_pim_contact*, *fnbl_pim_address* tables allow the Funambol PIM connector to query directly the DB in order to achieve the list of the new, updated, deleted items (see DAO layer specification)

11. Appendices

11.1. Appendix A – References

[1]