

**SemanticLIFE - Outlook Datafeed Module
Software Architecture Document**

Version 2.0

SemanticLIFE - Outlook Datafeed Module	Version: 2.0
Software Architecture Document	Date: 20/April/2005

Revision History

Date	Version	Description	Author
15/April/2005	1.0	Architectural description of Data-feed module of MS Outlook Data for the SemanticLIFE system	Hoang Huu Hanh
20/April/2005	2.0	Modification: Use Case View section, Sequence diagrams and refining some parts	Hoang Huu Hanh

SemanticLIFE - Outlook Datafeed Module	Version: 2.0
Software Architecture Document	Date: 20/April/2005

Table of Contents

1.	Introduction	4
1.1	Purpose	4
1.2	Scope	4
1.3	Definitions, Acronyms and Abbreviations	4
1.4	References	4
1.5	Overview	5
2.	Architectural Representation	5
3.	Architectural Goals and Constraints	5
3.1	MAPI Constraints	5
3.2	SyncML & Ssync4j Constraints	5
3.3	Standards and Used Technologies	6
4.	Use-Case View	6
4.1	Use-Case Realizations	6
4.1.1	Sync Engine Use case	7
4.1.2	Repository Use case	7
4.1.3	Outlook Data Session Use case	7
5.	Logical View	8
5.1	Overview	8
5.2	Architecturally Significant Design Packages	8
5.2.1	OL SyncClient Package	9
5.2.2	Prepare Package	9
5.2.3	Sync Package	9
5.2.4	Utilities Package	10
6.	Deployment View	11
7.	Implementation View	11
8.	Size and Performance	12
9.	Quality	12
9.1	Performance	12
9.2	Security	12
9.3	Portability	12
9.4	Reusability	12

Error! Bookmark not defined.

SemanticLIFE - Outlook Datafeed Module	Version: 2.0
Software Architecture Document	Date: 20/April/2005

Software Architecture Document

1. Introduction

This document provides a high level overview of the technical architecture for the Data-feed Module for Outlook Data in the SemanticLIFE system. It outlines the technologies that have been used for a data feed module with data sources are OL data including calendar, contacts, tasks and notes. This module will take changes in OL data, make the synchronization and upload XML-based messages of OL data to SemanticLIFE repository.

The document provides a high-level description of the goals of the architecture, the use cases support by the system and architectural styles and components that have been selected to best achieve the use cases. This framework then allows for the development of the design criteria and documents that define the technical and domain standards in detail.

1.1 Purpose

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

This document will provide an overview of technologies used and software architecture for Outlook Datafeed module for SemanticLIFE system. SemanticLIFE system is a PIM for storing and retrieving human lifetime information. In currently, the data feeds for the system are: personal data (calendar, contacts, tasks, notes), emails, browsed web pages, monitored processes and monitored file system. This document, as mentioned above, describes technical issues of datafeed module for personal data stored in OL (called OL data).

1.2 Scope

This software architecture document applies to the design of Datafeed Module of Outlook data for the SemanticLIFE system (SemanticLIFE project, Institute of Software Technology and Interactive Systems, Vienna University of Technology).

1.3 Definitions, Acronyms and Abbreviations

OL	Microsoft Outlook, http://office.microsoft.com/outlook
OL Data	Microsoft Outlook data includes calendar, contacts, tasks and notes (emails are not taken into account in this context)
MAPI	Messaging Application Programming Interface, http://msdn.microsoft.com/library/en-us/e2k3/e2k3/techsel_tech_13.asp
SyncML	Synchronization Markup Language, http://www.syncml.org
VS	Microsoft Visual Studio

1.4 References

- MAPI SDK, http://msdn.microsoft.com/library/en-us/exchanchor/htms/msexchsvr_mapi.asp
- SyncML Specification, <http://www.openmobilealliance.org/tech/affiliates/syncml/syncmlindex.html>
- The Sync4j Project, <http://sync4j.funambol.com>

SemanticLIFE - Outlook Datafeed Module	Version: 2.0
Software Architecture Document	Date: 20/April/2005

1.5 Overview

In the upcoming sections we will discuss the OL data feed module used in the SemanticLIFE Datafeed module. For simplicity we will refer to this application as OL Datafeed in the rest of this document.

In Section 2 the software architecture of the system will be explored and the necessary views to depict the software architecture are specified. Section 3 describes the architectural goal and constraints. In sections 4, 5, 6 and 7 (the Use-case view, Logical view, Deployment view and Implementation view) will be discussed respectively. In Sections 8 and 9 performance and quality issues will be discussed.

2. Architectural Representation

In the upcoming sections the system architecture will be viewed from different perspectives at a high level of abstraction that allows us to visualize, understand and reason about the architecturally significant elements and identify areas of risk that require more detailed elaboration.

The architecture of the reference application is represented following the recommendations of the Rational Unified Process (RUP). The system specifications have been divided into the following five views:

- **Use-Case View:** Describes the actors and use cases for the system, this view presents the needs of the user and is elaborated further at the design level to describe discrete flows and constraints in more detail.
- **Logical View:** This view describes the architecturally significant parts of the design model, such as its decomposition into subsystems and packages. And for each significant package, its decomposition into classes and class utilities. We will introduce architecturally significant classes and describe their responsibilities, as well as important relationships, operations, and attributes.
- **Deployment View:** Describes the deployment structures, by including known deployment methods. It will also describe the physical network (hardware) configurations on which the software is deployed and run. In this configuration we will indicate the physical nodes that execute the software, and their interconnections
- **Implementation View:** This view describes the overall structure of the implementation model, the decomposition of the software into layers and subsystems in the implementation model, and all architecturally significant components.

3. Architectural Goals and Constraints

OL Datafeed module is based on two main technologies, MAPI, SyncML (Ssync4j) in order extract data from OL database (including calendar, contacts, tasks, notes) and detect if there is any change from the last upload (sync) and then make synchronize the data and upload the new or changed data items.

3.1 MAPI Constraints

MAPI is not open source. However, it comes along with Microsoft products like MS Office System, MS Exchange, etc. So that it is free to use for developing.

3.2 SyncML & Ssync4j Constraints

SyncML and its open-source Java implementation are totally free for using and developing. Ssync4j is based on the client/server model. So that it requires a server side application called Ssync4j Server (running on Tomcat or JBoss)

SemanticLIFE - Outlook Datafeed Module	Version: 2.0
Software Architecture Document	Date: 20/April/2005

3.3 Standards and Used Technologies

The OL Datafeed should use a set of standard technologies for implementation. These standards are those that are based on open source applications. The employed technologies are as follows:

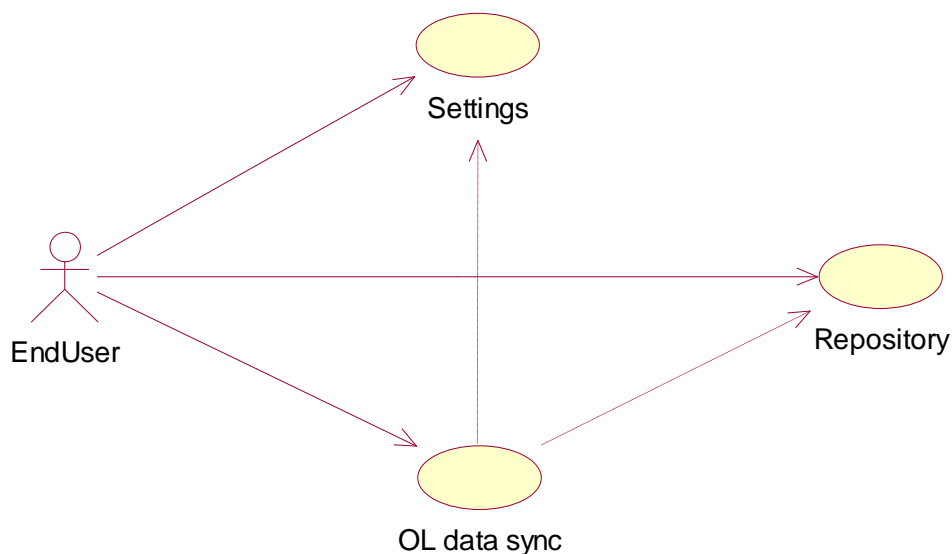
- **MAPI:** Messaging Application Programming Interface is the heart of Microsoft's Messaging programs. Extended MAPI serves following main purposes:
 - It's the programming interface used to write components that connect to different mail servers, provide access to custom address books and provide rich storage facilities.
 - You can create addins for Outlook, Exchange and Windows Messaging that extend the functionality of those clients
- **SyncML and Sync4j:** SyncML is a standard for synchronization in mobile world. It supports many types of synchronization. Its Java implementation is Sync4j provides API and the environment for synchronization of data from devices such as Outlook, calendar and contacts of handheld devices. These technologies are used to detect changed items in Outlook data concerning SemanticLIFE system.
- **MSXML:** Microsoft XML parser will be used for XMLizing messages and also used for reading/writing configuration file of OL Datafeed module.
- **Visual Studio:** Microsoft Visual Studio is a powerful platform to develop applications. Because of used libraries, it will be used as development tool.

4. Use-Case View

This view presents the users perception of the functionality provided by OL Datafeed. These use cases were included in the exercise specifications and hopefully captures all requested features.

Note that this section provides the context for, and scope of the rest of the document. Putting aside overriding architectural constraints outlined above all development (in terms of design and content documentation) has been done in support of one or more of the following use cases.

4.1 Use-Case Realizations



SemanticLIFE - Outlook Datafeed Module	Version: 2.0
Software Architecture Document	Date: 20/April/2005

4.1.1 Settings Use case

In this use case the end user who is considered as use case actor will change the settings for sync session and uploading session as well. The settings (so-called utilities) include scheduler, configuration file, and synced data uploader.

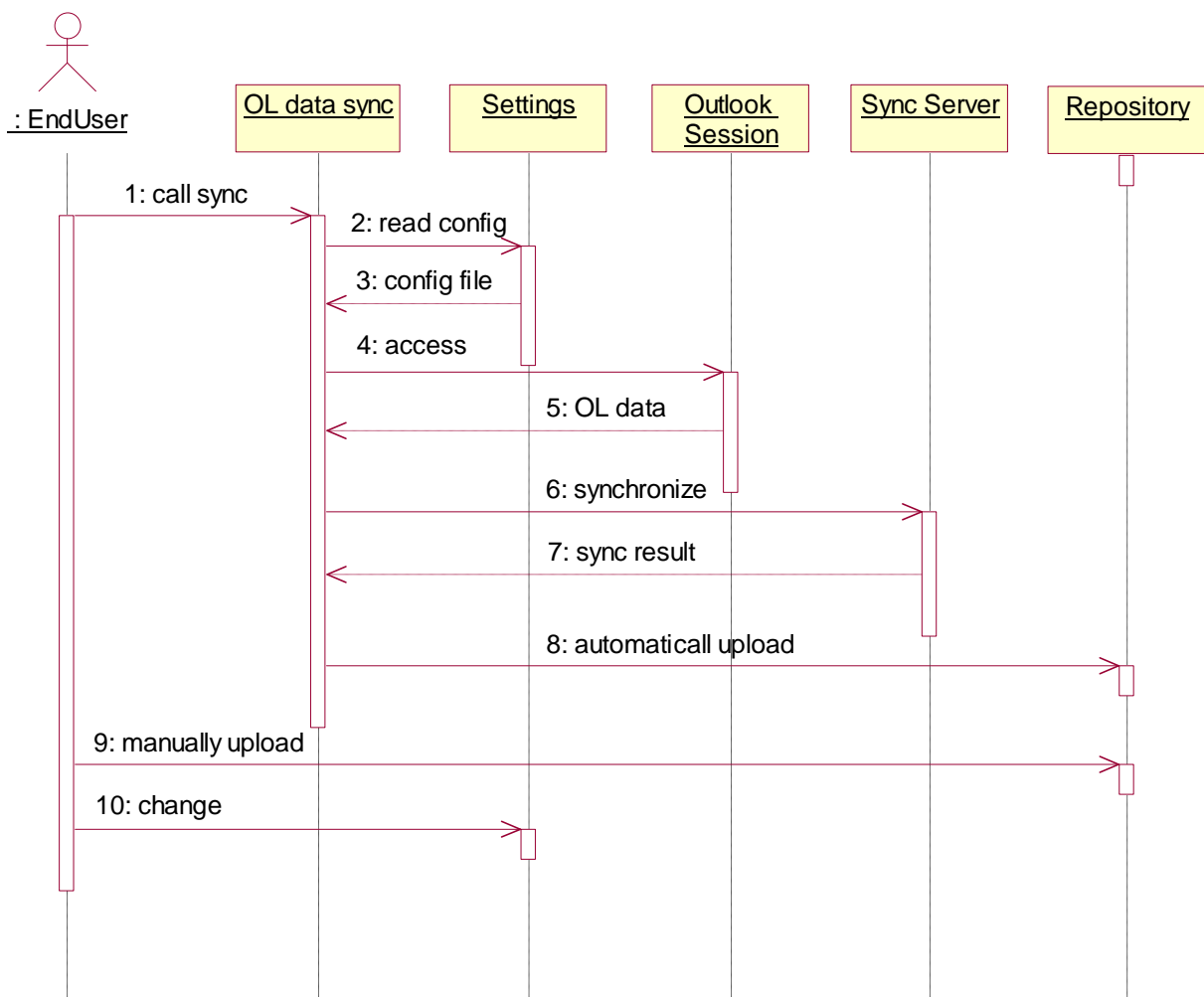
4.1.2 OL data Sync Use case

In this use case the end user who is considered as use case actor will make synchronization between current OL data and Sync4j server which maybe contain synchronized items already. The new, changed or deleted items will be detected. After that, these items would be uploaded to the repository automatically. This use case relates to settings (for filtering, scheduling) and Outlook database as well.

4.1.3 Repository Use case

In this use case end user will upload detected items (XML files) after synchronization into the repository of SemanticLIFE system.

The following diagram shows the interactivity between basic components to fulfill the requirements of this use case:



SemanticLIFE - Outlook Datafeed Module	Version: 2.0
Software Architecture Document	Date: 20/April/2005

5. Logical View

This section describes the architecturally significant elements of the Datafeed module. The system is decomposed into some sub-systems, to realize the requirements of the above mentioned use cases.

5.1 Overview

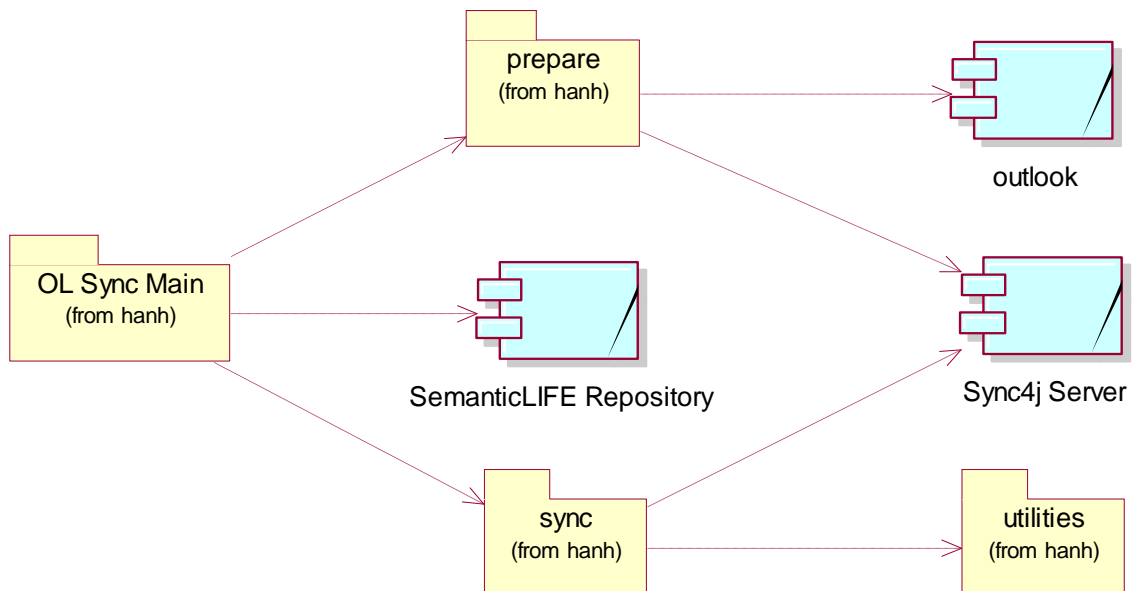
The design model is depicted as following diagram. These generated files are organized into one package which contains 3 sub-packages; main package is OL SyncClient, one sub-package for preparing synchronization, one for making synchronization and the other one for utilities such as config/filter, scheduler, uploader. The package names are “OL Sync Main”, “prepare”, “sync” and “utilities” respectively.

The most interesting package is “sync” package which includes all the needed modules for synchronizing. This session will use sync lib and access to the sync server using Sync4j to retrieve the sync results. During this transaction, it will also use parameters of configuration file in “utilities” to filter the results.

The “prepare” package will access Outlook and the sync server to prepare needed sources for synchronization.

The “utilities” package contains some facilities for sync. It includes a function for configuration file (a XML file) used for sync, the scheduler for automatic sync and the uploader for publishing result XML messages to Semantic repository

Finally the “OL Sync Main” package will integrate all packages into an friendly user interface. From this point, the user can access and use functions to synchronize his/her OL data sources.

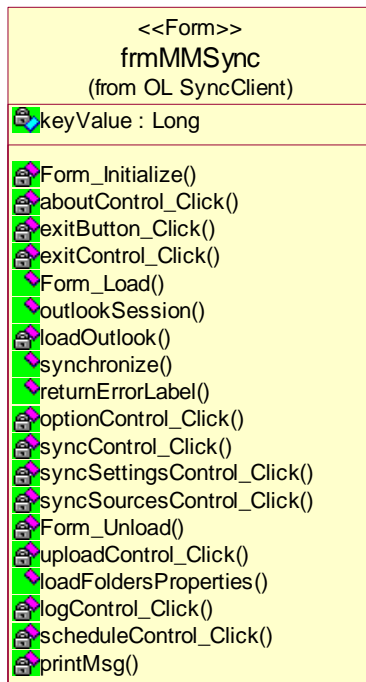


5.2 Architecturally Significant Design Packages

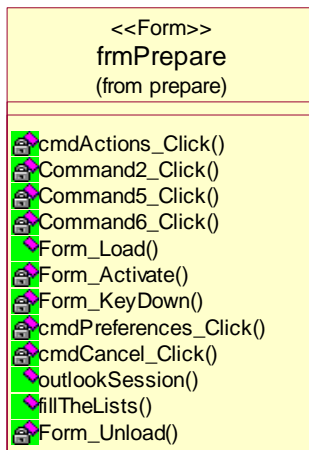
In this section we will describe the details of significant packages including the description of important classes.

SemanticLIFE - Outlook Datafeed Module	Version: 2.0
Software Architecture Document	Date: 20/April/2005

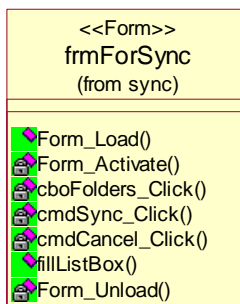
5.2.1 OL Sync Main Package



5.2.2 Prepare Package

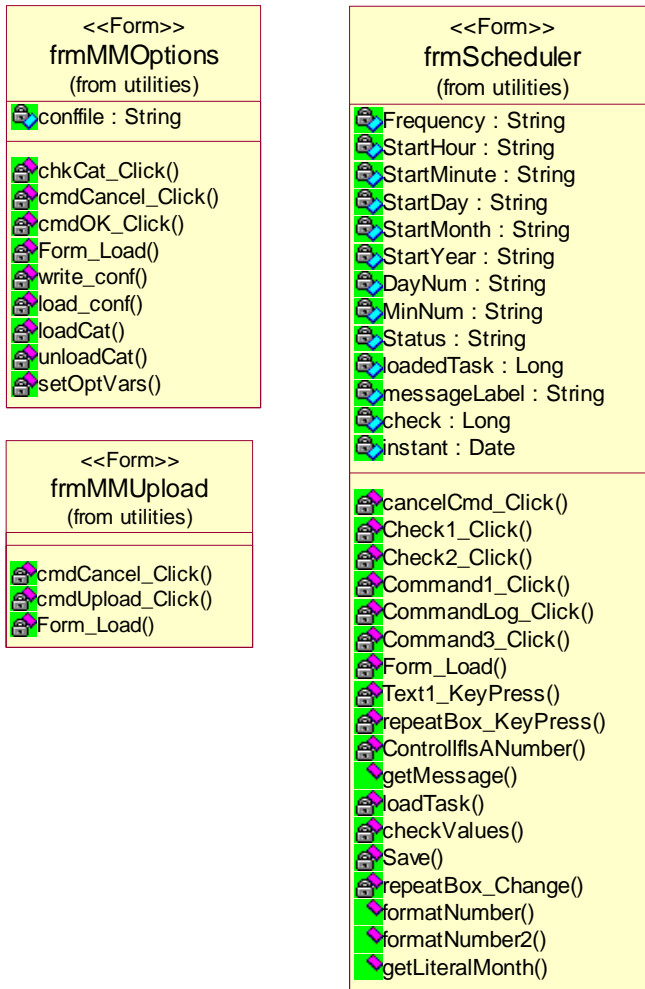


5.2.3 Sync Package

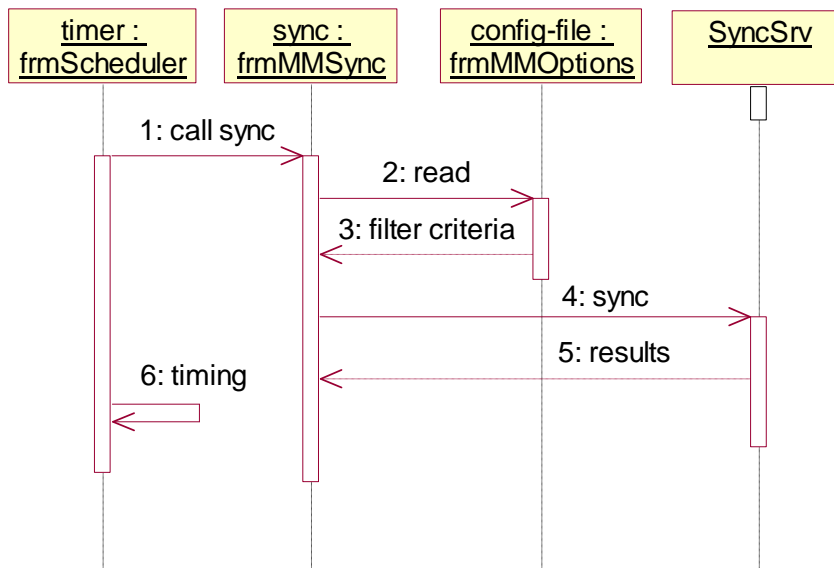


SemanticLIFE - Outlook Datafeed Module	Version: 2.0
Software Architecture Document	Date: 20/April/2005

5.2.4 Utilities Package



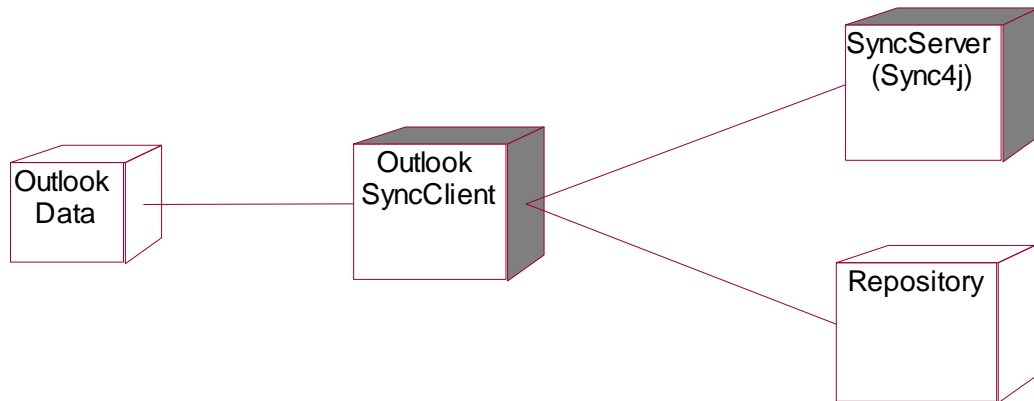
The following diagram shows the interactivity between this class and other classes to perform the synchronization:



SemanticLIFE - Outlook Datafeed Module	Version: 2.0
Software Architecture Document	Date: 20/April/2005

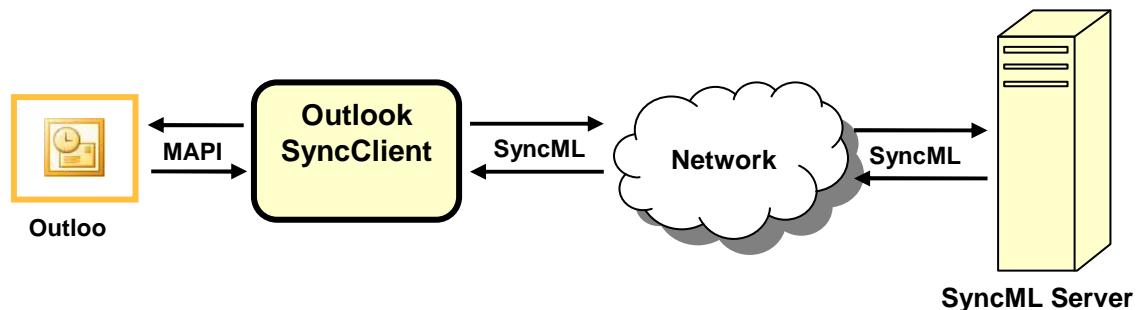
6. Deployment View

The system is composed of client side and server side components. The server component is built on Sync4j. The client computer will be deployed at client side. This client should have access to Outlook and Sync4j to perform synchronization and finally uploading the synchronized results (XML files) into the repository.



7. Implementation View

This section describes the overall structure of the implementation model, the decomposition of the software into layers and subsystems in the implementation model, and some architecturally significant components. The overall view of system components are shown in the following diagram:

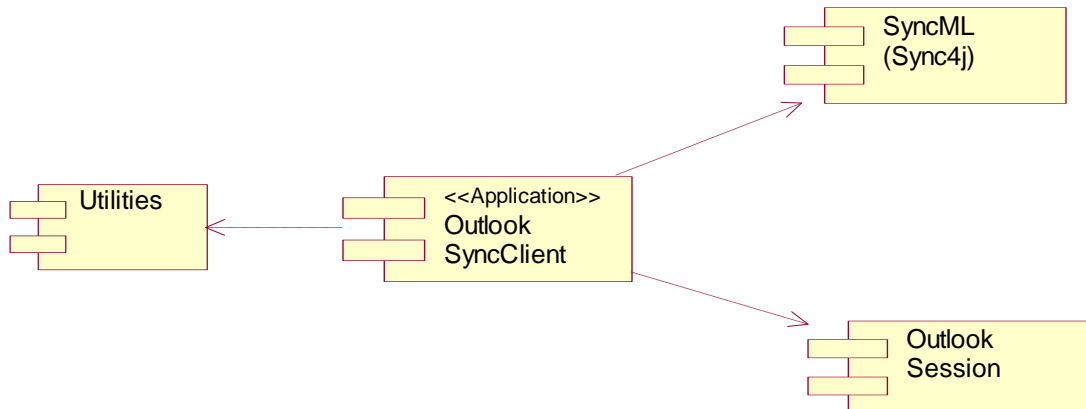


MAPI provides the facility to access OL data. Deployment Outlook Session is based on this API.

Sync4j provides a bundled Java-based implementation of SyncML standard. Sync4j Server will run on Tomcat or JBOSS. A set of APIs for Java, C++ is also provided. The transaction protocols between Sync clients and Sync4j Server base on HTTP and SyncML.

The following shows the components from implementation point of view:

SemanticLIFE - Outlook Datafeed Module	Version: 2.0
Software Architecture Document	Date: 20/April/2005



8. Size and Performance

This OL Datafeed module is run by a user. The filtering criteria and proposed destination of the application can be set in the configuration file. The module will access to Outlook data and make synchronization with the sync server. This module performs well locally or through intranet, however when the system is used in the real world, the performance of it depends on the nature of the networks through HTTP transaction protocol.

9. Quality

The overall system quality could be overseen from different aspects. In this section we will discuss some important quality issues like performance, security, portability, functionality and reusability.

9.1 Performance

Performance of OL Datafeed module depends on synchronization speed. The speed of synchronization in new version (Sync4j Server 4.0.8) is good enough. However during the sync session, it uses HTTP protocol to make the transaction protocol between sync client and sync server so that the performance is also depends on quality of the network (computer network or mobile network).

9.2 Security

Security is an important issue in implementation of application using HTTP protocol and there are well-known methods which apply security policies to it. In our case the security is not our primary concern since we do a simple transaction. Should the system be extended to support commercial issues, the security issues come to play.

9.3 Portability

Portability issues are about abstracting the interfaces that a business application uses to communicate with the underlying system upon which it is running. This application is developed basing on Microsoft platforms such as Office (Outlook), Win32 OS and Win 32 development tool (Visual Studio). The application can run effectively on Microsoft operating systems and any MS Office version. So the application's portability is acceptable.