A Software Library for SyncML Server Applications

JiYeon Lee[*], Hoon Choi**
Dept. of Computer Engineering, Chungnam National University
220 Kung Dong, Taejeon, 305-764, Korea
E-mail : {eunbi, hchoi}@ce.cnu.ac.kr

Abstract

The SyncML, the standard synchronization protocol, supports the synchronization of various application services such as an address book, a calendar. Even with this standard protocol, application developers usually spend a long time implementing service specific logics and databases. This paper designed and implemented a software library of common functions needed to implement SyncML applications so that developers of SyncML server code can save time and effort using this library. We implemented a SyncML server with two application services, vCard and vCalendar, using this library. The test of the library and the server was carried out by the SyncML conformance test specification..

1. Introduction

Data synchronization is an arbitration operation to make replicated data on different devices consistent with each other by resolving data conflicts. The mobile industries including IBM, Lotus, Motorola, Nokia, Palm, Psion, Starfish Software organized the SyncML (Synchronization Markup Language) Initiative in order to develop an open standard for data synchronization mechanisms which can be applied to heterogeneous platforms, networks and application services[1][2]. They released the SyncML specification 1.0 in December 2000 and the latest version 1.0.1 in June 2001[3][4][5][6]. The SyncML defines the common language and the protocols of data synchronization.

A typical application which benefits by the SyncML protocols is the PIMS (Personal Information Management System) service such as an address book or a calendar, but the protocols will support a variety of mobile application services. There are many works for software engineers when they build an application server from scratch, or they add a new application to an existing server; implementing SyncML protocols, creating and updating a database, adding a new service logic and changing many source codes at the server side and the client side. The SyncML protocols define only the standard language to represent the synchronization information and the message exchanging procedure. So, developers of application software are required to implement application specific part as well as the SyncML protocols. Despite the standardization of lower level protocols, it takes for developers a long time to implement the application specific part for each application and this job is difficult and also error-prone. If APIs (Application Programming Interfaces) are available which can be commonly used for all synchronization applications, developers are able to work with the APIs to develop SyncML application services easily in a short time period.

This paper introduces a software library of common functions needed to implement SyncML applications so that developers of SyncML server code can save time and effort using this library. In order to design the Sync Library, first we implemented a SyncML server with two application services, vCard and vCalendar. Then, we extracted sharable functions of the server code and implement them in a form of a library with generalized APIs. This library aims to be used with an automated code generation system that we are currently working on.

We briefly introduce the architecture and functions of our CNU SyncML Server 3.1 in Section 2. Then, we describe the functions and APIs of the Sync Library in Section 4. Test of the library is also mentioned.

2. CNU SyncML Server

We implemented a data synchronization server, named the CNU SyncML Server 3.1, based on the standard protocol specification of the SyncML Initiative. They are able to process ADD, ALERT, COPY, DELETE, GET, MAP, PUT, REPLACE, RESULTS, SEARCH,

* University Research Program supported by Ministry of Information & Communication in South Korea
** This author was supported by SOREC(Software Research Center) of KOSEF in Chungnam National Univ. SOREC is a Regional Research Center Designated by Korea Science and Engineering Foundation.

SEQUENCE, STATUS, SYNC commands and six synchronization scenarios such as Two-way sync, Slow sync, One-way sync from client only, One-way sync from server only, Refresh sync from client only, Refresh sync from server only. Application services that we implemented include a vCard formatted address book service and a vCalendar formatted calendar service.

The CNU SyncML Server 3.1 has the functional architecture shown in Figure 1.

The Server Application provides an interface for a user or the application administrator to modify contents data at the server. The Sync Adapter passes messages from client devices to the SyncML Toolkit. The Sync Adapter also passes the parsed data from the Toolkit to the Sync Agent. The SyncML Toolkit encodes and decodes SyncML messages. We used the reference implementation code from the SyncML Initiative for the SyncML Toolkit[7]. The Sync Agent implements the synchronization protocol[4] and the representation protocol of the SyncML[3]. Therefore, it is application independent and can be commonly used for all SyncML applications. On the other hand, the Sync Engine implements application dependent part such as handling service policies, user authentication, and defining resolution rules in case of data conflict. The Session Manager keeps temporary, session related information that needs to be maintained during a session in the server. Lastly, the Open DB Interface implements interfaces to access data in database[8].

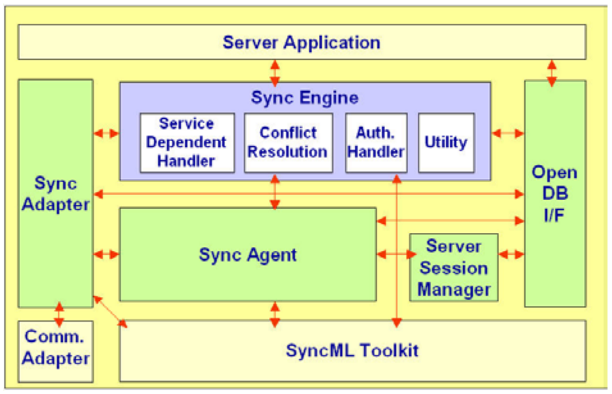


Figure 1. Frames of the CNU SyncML Server 3.1

All the frames of the CNU SyncML Server except the Session Manager were implemented into DLLs (Dynamic Link Library). The Session Manager was implemented into an independent process because implementing it into a DLL will lose the session information once the DLL is unloaded. The communication with other frames in the server is done by RPC (Remote Procedure Call). We used HTTP(Hyper Text Transfer Protocol)[6] to communicate with remote SyncML client devices and JNI (Java Native Interface) to load DLLs.

The SyncML can be used for many application services such a calendar, an address book, an email, and a text messaging service.

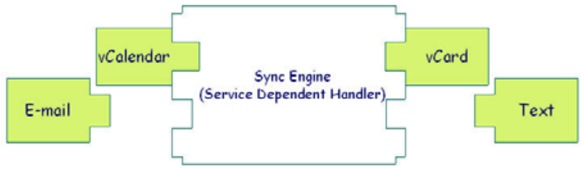


Figure 2. Service Dependent Handler

We designed the server so that the code that is dependent on a specific application is separated from the application independent code. This is for application developers to reuse service independent part when he adds a new application. The developers need to change only the Sync Engine. This architecture results in the better extensibility of the server, reduced costs and fast development of new application services (Figure 2).

3. Sync Library

Even though we separate the service dependent part from the independent ones to minimize the effort of adding a new service, implementing a new service in the Sync Engine still requires a lot of work and expert level knowledge about the server code such as adding a database table related to application service, adding APIs to interface between the Sync Engine and database, adding APIs to interface the Sync Engine with the Sync Agent and so on.

APIs of the Sync Engine and database require different input parameters, data structures and database fields specific for each application. Each application requires its own functions implemented in the Sync Engine. But, the basic role and usage of the functions are very similar from one application to other.

Therefore, we generalized the functions in the application service dependent parts and implemented them into a software library called the Sync Library.

3.1. Implementation of the Sync Library

We classified the application dependent part into two groups, one is for database and the other is source code that performs data synchronization. Service dependent part in the database includes an application identifier and database tables specific for the application. In source codes, the dependent part includes the application data structures, the interface code to database in order for storing the application contents data, and the Sync Engine.

We designed and implemented service dependent part and other common and sharable functions into the Utility module and the Core module of Figure 3.

The Utility module provides the general utilities. The Core module is composed of the functions, which is similar to the Service Dependent Handler module of the CNU SyncML Server 3.1. The main roles of the core module are command processing and data processing. The command processing is to process ADD, REPLACE, DELETE, COPY commands and to find contents data. The data processing is to encode, decode the vCard or vCalendar formatted data and to compare the data field by field.

For implementation of the Sync Library, we defined the Utility class, the Core class for each module. Table 1 lists the methods and their operations of the Core class.

Table 1. The methods of Core class

Method	Description
cmdAddEntry	It adds the contents data to database with new GUID of an application service
cmdReplaceEntry	It replaces the contents data of an application service
cmdDeleteEntry	It deletes the contents data of an application service
cmdCopyEntry	It makes a copy of an entry of an application service
cmdFindEntry	It retrieves the entry with GUID of an application service
datCompare	It compares the contents data of two entries field by field of an application service and returns the results
datEncode	It encodes the contents data into each application format and returns the results
datDecode	It decodes the encoded data into internal data format and returns the results

The Utility class has many general functions; to make a token for an application service, to control a general string, to generate a string including certain information, to find an interesting information from a string, to convert local time format into UTC(Universal Time, Coordinated) time format and so on. Table 2 describes some of the methods and their operations of the Utility class.

Table 2. The methods of the Utility class

Method	Description
utParseURI	It gets an application type and LUID from a URI
utBuildURI	It generates a URI with an application type and LUID
utGetdbIDfromGuid	It splits the source address of a message into GUID and an application identifier
utMakeUTC	It converts a local time string into a UTC time string
utChopNull	It chops NULL character from a string
utGetToken	It gets a delimiter and a token value from a given string
utTokenize	It makes a token for each application
utLog	It writes the execution status and results

The Sync Library was implemented as a static library using C++. A server developer can easily use the APIs and functions by linking the library to the server.

3.2. Implementation of the CNU SyncML Server using the Sync Library

We implemented a new version of the CNU SyncML Server 3.1 using the Sync Library to show the correctness of the library operations. The new server also supports synchronization of an address book and a calendar. The following Figure 3 shows the framework of the server based on the Sync Library. The Sync Agent interfaces with the Sync Engine in the same way as the version 3.1 does. The Service Dependent Handler module of the Sync Engine is implemented by using the Sync Library APIs and supports to synchronize a vCard formatted address book and a vCalendar formatted calendar like the server version 3.1.

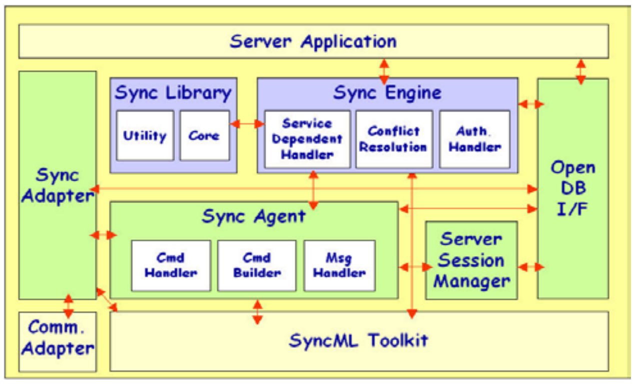


Figure 3. Framework of the SyncML Server based on the Sync Library

There is still the Service Dependent Handler module in the new implementation. But this module is different from the Service Dependent Handler of the CNU SyncML

Server 3.1. The new Service Dependent Handler is mostly coded with calls to APIs of the Sync Library, so is much compact in size. A developer can save much of programming effort by using pre-defined library APIs.

We verified the Sync Library by testing the server based on the Sync Library according to the conformance test specification of the SyncML Initiative[9][10]. The SyncML conformance test scenarios consist of exchanging messages that include various commands such as ADD, REPLACE, DELETE and COPY between clients and the server for each application service. We also used our own test scenarios for synchronization tests. Both of the two application services; an address book and a calendar are tested intensively. We compared and analyzed the test results from the server with the Sync Library according to the conformance test specification and with the results from the CNU SyncML Server 3.1.

Table 3. Results of 12 test cases of the conformance test

Test		Result	Note
①	Two-way with Client & Empty Server	pass	○
②	Two-way with Client Add & Server Add	pass	○
③	Two-way with Client Replace & Server Replace	pass	○
④	Two-way with Client delete & Server Delete	pass	○
⑤	Two-way with Client Add (shows empty server sync data)	pass	○
⑥	Two-way with Server Add (shows empty client sync data)	pass	○
⑦	Two-way with Server with large amount of data (shows multiple message)	pass	○
⑧	Two-way with Client with large amount of data (shows multiple message)	pass	○
⑨	Two-way with Server responding busy	pass	○
⑩	Two-way with Server not responding		Client only
⑪	Two-way with communication broken during sync	pass	○
⑫	Two-way Slow Sync	pass	○

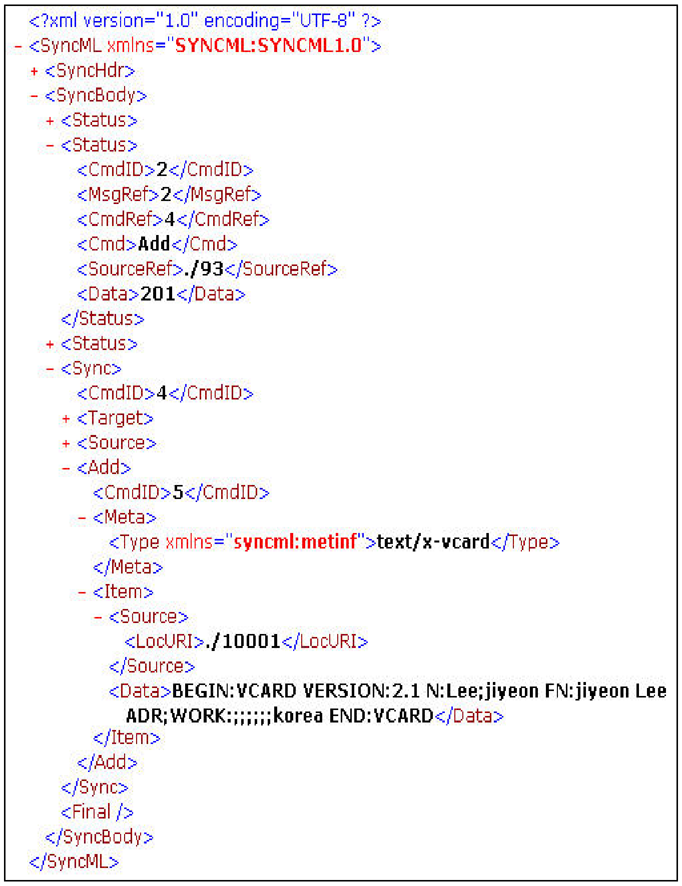
Figure 4 shows a part of the server message. This is for the case that a server processes the requests from a client and then sends a client the result status with the data changed by the server. We confirmed that the two servers have the same results.

4. Conclusion

We redesigned the code for application service-dependent parts of a SyncML server and generalized the functions into a software library. Interface of the library is defined as APIs to use when we build a SyncML server or add a new application to the server. We re-implemented our CNU SyncML Server 3.1 using the Sync Library in order to verify the functions of the Sync Library. The server with Sync Library worked well and passed all the test cases of the SyncML conformance test.

Using the Sync Library, developers do not need to implement functions that already built in the library, therefore they can save a lot of time and effort. This will contribute efficient and economical development of new applications.

As a follow-on research, we are currently working on an automated tool that uses the Sync Library and generates source code of application service-dependent parts. With this tool, we hope that development process of a new service becomes even faster and easier.



```
  <?xml version="1.0" encoding="UTF-8" ?>
- <SyncML xmlns="SYNCML:SYNCML1.0">
  + <SyncHdr>
  - <SyncBody>
    + <Status>
    - <Status>
        <CmdID>2</CmdID>
        <MsgRef>2</MsgRef>
        <CmdRef>4</CmdRef>
        <Cmd>Add</Cmd>
        <SourceRef>./93</SourceRef>
        <Data>201</Data>
      </Status>
    + <Status>
    - <Sync>
        <CmdID>4</CmdID>
      + <Target>
      + <Source>
      - <Add>
          <CmdID>5</CmdID>
        - <Meta>
            <Type xmlns="syncml:metinf">text/x-vcard</Type>
          </Meta>
        - <Item>
          - <Source>
              <LocURI>./10001</LocURI>
            </Source>
            <Data>BEGIN:VCARD VERSION:2.1 N:Lee;jiyeon FN:jiyeon Lee
              ADR;WORK:;;;;;;korea END:VCARD</Data>
          </Item>
        </Add>
      </Sync>
      <Final />
    </SyncBody>
  </SyncML>
```

Figure 4. A part of the result message from a server

5. References

[1] SyncML Initiative, http://www.syncml.org

[2] SyncML Initiative, Building an Industry-Wide Mobile Data Synchronization Protocol, SyncML White paper, Mar. 20, 2000

[3] SyncML Initiative, SyncML Representation Protocol version 1.0.1c, June 15, 2001

[4] SyncML Initiative, SyncML Synchronization Protocol version 1.0.1, June 15, 2001

[5] SyncML Initiative, SyncML Architecture, version 0.2, May 10, 2001

[6] SyncML Initiative, SyncML HTTP Binding, version 1.0.1, June 15, 2001

[7] SyncML Initiative, SDA2 Specification Version 0.2, Aug. 21,2000

[8] Byung-Yun Lee, Gil-Haeng Lee, JinHyun Cho, SooHee Ryoo and Hoon Choi, "SyncML Data Synchronization System based on Session Manager," Journal of KISS, Vol.8, No. 6, pp.647-656, Dec. 2002.

[9] SyncML Initiative, SyncML Conformance Test Suite version 0.2, 2001.1.15

[10] SyncML Initiative, SyncML Manual Test Cases version 0.3, 2001.2.2