Funambol™ Data Synchronization Server Overview

Summary

Over the last few years, wireless data networks and mobile devices capable of sophisticated communication have grown tremendously. Increasingly, users are demanding that their applications and data are available on their desktop, laptop and a mobile even when disconnected. The combination of an application server and Funambol mobile application server, as the mobile application server, enables the synchronization of data among all of these platforms via SyncML. SyncML is a protocol implemented in virtually all shipping mobile phones allowing the synchronization of data among a heterogeneous set of devices and applications. This article provides an introduction to SyncML, its use, and integration of the Funambol Data Synchronization (DS) mobile application framework.

Introduction

As with many technologies, various vendors provide similar applications, but the applications are sometimes incompatible between vendors. This was the case with vendors of mobile devices. Today, the Palm handheld and the Pocket PC are still incompatible when it comes to data synchronization. Both require their own desktop software to synchronize between device and the desktop. In order to converge to a single standard, a group of vendors combined efforts to start the SyncML initiative in 2000. This initiative was aimed at the definition and promotion of a universal synchronization technology based on the de facto standard XML. They defined the SyncML protocol that enabled synchronization of data and in particular PIM -- contact and calendar -- data. Today, this protocol is supported by virtually every handheld device, ranging from simple phones to smart phones and handheld computers. Although devices can synchronize with each other, SyncML is mostly a client/server oriented protocol that uses HTTP as a transport. The Funambol DS server extends existing application servers with the SyncML protocol and builds upon the HTTP protocol and other services that an application server usually provides.

SyncML Technology

SyncML is based on the sharing of changes between two applications and using those changes to update a local data set. The most well known example is the synchronization of PIM data between a desktop and a handheld device. SyncML goes even further by allowing any kind of data to be synchronized between many different sources of data. However, the most common deployment is that every device shares its changes with a server and the server always contains the latest up-to-date data set to be used for updating other clients. This provides an easier use of synchronization technology, since every device only needs to know about a single server and how to contact it.

During synchronization, conflicts can arise if the server detects a change of an item on both the device and the server since the last update. To resolve such a conflict, various possibilities exist, for instance, server changes could override device changes.

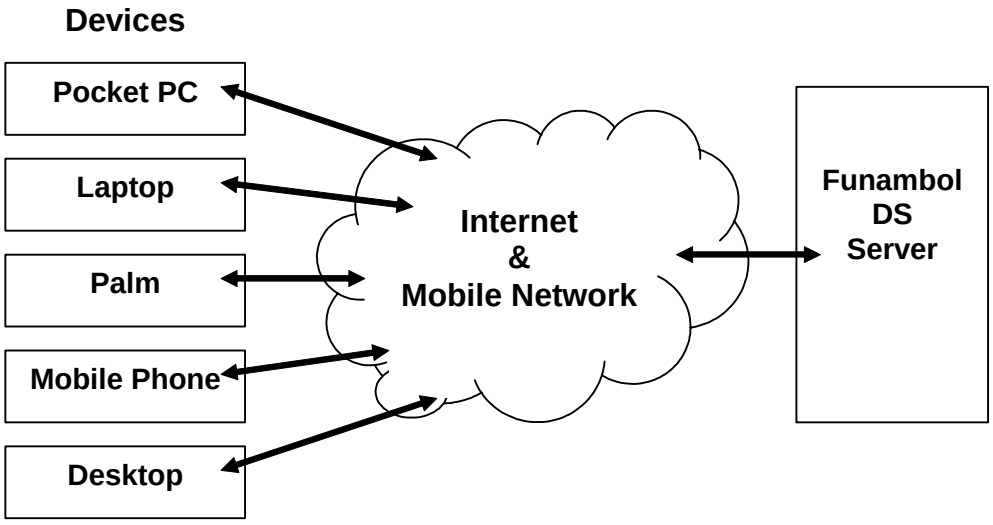

Figure 1. SyncML Deployment

The above figure provides an overview of SyncML. On the left, various devices, from handheld devices to desktops, are shown. These communicate using the SyncML protocol via the Internet and/or mobile networks to the synchronization (DS) server. All devices synchronize their set of data against the sync server. This is achieved by a 1:1 synchronization of the devices with the server. The server keeps track of what data is updated in which device.

SyncML Protocol

SyncML is based on XML but it has multiple bindings to a transport protocol. For instance, the transport protocol may be HTTP or OBEX (Object Exchange Protocol), but regardless of the transport protocol, SyncML does not change.

A SyncML session consists of four phases:

1) The server alert phase, that is optional and may be considered a pre-initialization phase. Its primary purpose is to allow a server to inform a client that it should start a synchronization session.
2) The initialisation phase is started by the client (mobile device) to open a SyncML session to synchronize its data with the server. This phase handles the authentication of the client, authorization to access the databases, exchange of capabilities, synchronization content and type and a conformity check of the synchronization anchors.
3) The data exchange phase performs the actual data transfer. Normally, the client starts to send its changes after which the server processes it, detects and resolves conflicts and returns its changes.
4) The completion phase that is used to tell the server which mappings have been done, agree on the sync anchors and acknowledge the sessions for closure.

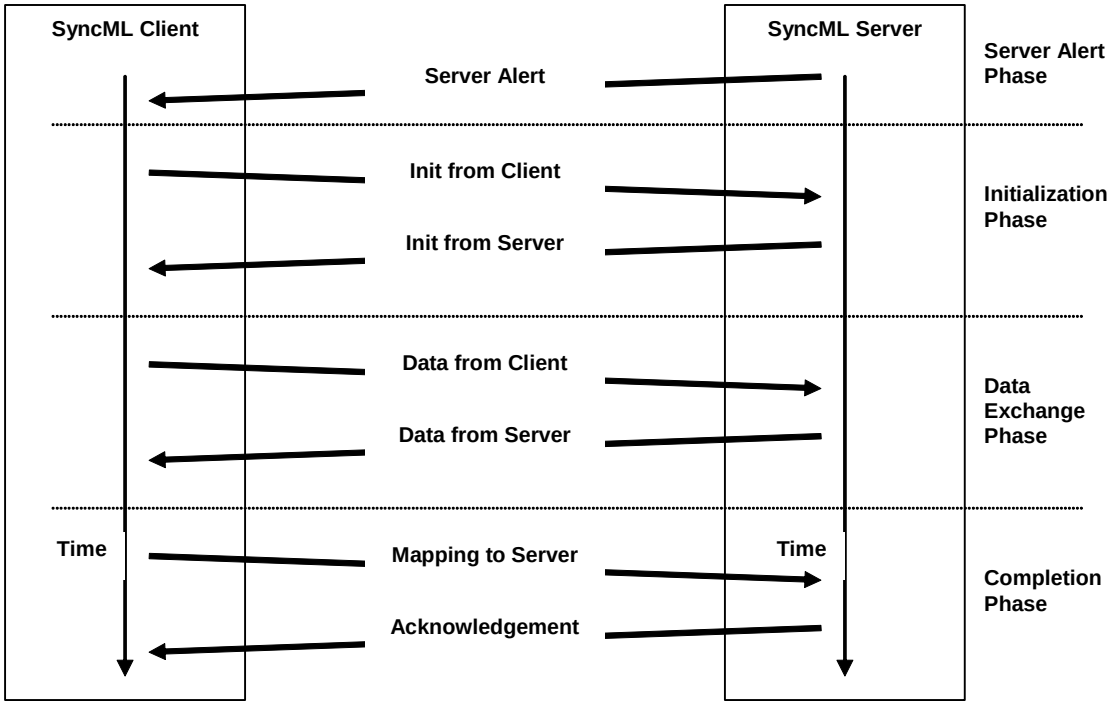

Figure 2: SyncML Communication Phases

The above picture provides a time based order of the phases. The behavior is according to the request-response paradigm, but may consist of multiple message exchanges per phase. This allows resource-limited mobile devices to accept smaller messages, but multiple messages to compose the logical request of a particular phase.

Funambol DS Server Architecture

Tomcat provides the application framework for the Funambol DS Server. It provides the communication capabilities for HTTP, the session handling, database pooling and many other features. By using existing application servers like Tomcat or JBoss, the Funambol DS Server is able to focus only on the tasks of the SyncML application protocol. This is sound example of re-use of existing technology and eases integration of SyncML in existing environments. In many organizations providing services, application servers are already used and even the maintenance and installation allows re-use of skills.

The Funambol DS Server architecture is build around the sync engine that provides the core features of the server. The sync engine processes internal representations of SyncML messages received from clients and constructs an internal representation of the SyncML message for the client response. The composition of the Funambol DS module for an application server is illustrated in Figure 3.

For most developers of mobile applications, the sync engine is not of concern; the developer will only make modules based on APIs provided by the sync engine. The most common example is the implementation of a SyncSource that interacts with the sync engine and the data store of your application. This is explained In the next section.

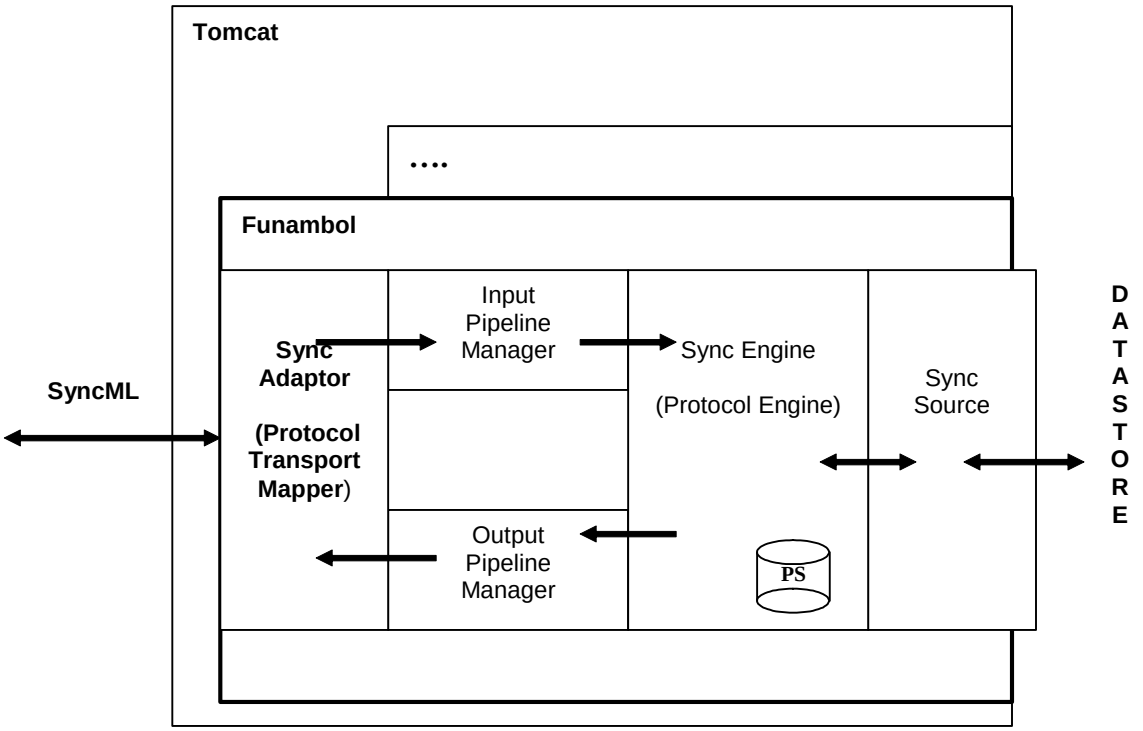

Figure 3: The Funambol DS Server Architecture

Figure 3 depicts the Funambol DS Server architecture within Tomcat. The Funambol DS architecture consists of six main components:

1. The sync adaptor or transport mapper that receives SyncML messages from HTTP and converts them into an internal representation that will be sent with HTTP.
2. The input pipeline manager allows additional processing of the incoming SyncML message. In general it checks if the internal representation is correct in order to be processed by the sync engine.
3. The output pipeline manager is similar to the input pipeline manager except it is for the other outgoing direction.
4. The sync engine is the core of the Funambol DS and provides the application functionality for the SyncML protocol.
5. The sync engine has access to a persistent store in which it stores information that it maintains between sessions regarding the state of synchronizations of devices.
6. The sync source is used to access the data store to which a mobile device synchronizes its data.

The Sync Adaptor

SyncML messages are XML based that may either go as-is over the wire or encoded with the WBXML format. This is specified by the mime-type, application/vnd.syncml+xml or application/vnd.syncml+wbxml, that is a header in the HTTP-request. The main task of the SyncAdaptor is to interact with the HTTP transport layer and to convert a SyncML message back and forth to the internal representation.

Normally, a SyncAdaptor does not require any customisation or integration effort. This is a protocol specific part and determined by the devices that access the server.

Pipeline Manager

The two pipeline managers are equal in principal. Their only difference is that one is for incoming messages and the other for outgoing messages. The pipeline manager allows additional processing of the SyncML message. The following are examples of processing can be implemented:

- Fixing message errors of known buggy devices that require corrections
- Performing pre-engine checks to block incoming messages that you do not want to process
- Logging of device capabilities devices to keep better track of them

This interface does normally not require any customization, but for deployed environments this provides the "hot" deployment mechanism of corrections for the server i.e. Synclets.

The Sync Engine

The sync engine is the core of Funambol DS providing the application functionality for the SyncML protocol. The engine handles the three phases: initialization, data exchange and finalization. In particular, one can find here functionality like authorization, change detection, conflict resolution, and id mapping. The functions that a user can change are authorization and conflict resolution.

Authorization

Every application needs authorization. The SyncEngine performs the checks for the authentication of users accessing the server. By default, Funambol DS comes with an authentication mechanism that checks username/password combinations in the database where Funambol DS stores other persistent information.

This authorization mechanism is implemented by an Officer. By default it verifies the username/password with a database (as part of the persistent store), but in many cases you many want to integrate Funambol DS with existing systems and/or user databases. That is considered a typical integration task.

Conflict resolution

SyncML performs the synchronization between two sets of data. The server retrieves from both sides, i.e. the client and the server, the changes which may indicate an item is new, updated or deleted. The Funambol DS Server processes this and constructs the required sets of instructions for both client and server to finish with an equivalent set of data. In simple terms, this means taking the change from one side and sending them to the other side. However, it may be possible that on a specific item, actions were done on both sides. In that case, the server will use conflict resolution to correct the data depending on the chosen strategy.

Changing of default conflict resolution rules is considered a typical integration task and/or definition of the application behavior. This interface is provided as Sync Strategy and can be configured.

The SyncSource

The sync source is the access to the data storage with which a mobile device synchronizes its data. A sync engine can maintain multiple SyncSources. Also, multiple syncsources can act upon the same data set. This is useful if the data items retrieved in the SyncSource require additional processing.

A typical integration project in an existing environment consists of the integration with the data storage. To provide SyncML access to data storage, one must implement a SyncSource (or connector). The definition of the SyncSource is hiding all SyncML related protocol issues and in practice, one should only deal with the content and key of data items besides a few well defined operations, like get/create/update/delete.

Conclusion

Funambol DS enables new usages of application servers and deployment of mobile applications and services. The generic web-based services such as HTTP, session handling, database access, and XML parsing that any application server provides make it is an ideal framework for SyncML applications. This allows Funambol DS to focus on the SyncML layer only. The Funambol DS platform is a complete framework that allows you to focus on mobile application development. In-depth understanding of SyncML is not required. This reduces total cost of ownership and decreases time to deployment.

About Funambol

Funambol is the mobile open source company. Funambol's Mobile Application Server offers "push" email, multimaster PIM synchronization, and management facilities for mobile devices. Funambol, formerly known as Sync4j, is an open source development platform for mobile applications that has been downloaded more than any other wireless middleware product – 750,000+ times. The commercial version has been deployed at wireless carriers, Fortune 100 enterprises, hardware OEMs/ODMs and ISVs including customers such as Computer Associates. Funambol is headquartered in Redwood City, California with a development center in Italy. For more information, please visit http://www.funambol.com.

Worldwide Headquarters
643 Bair Island Road, Suite 305
Redwood City, CA 94063 USA
(650) 701-1450 x105

Europe
Via dei Valtorta 21
20127 Milano Italy
+39 02 2614 5383