SyncML Overview

Noel Poore, Psion Computers PLC

Data synchronization is a field of growing importance. As the number of mobile devices increases rapidly in the next few years, more and more data is going to be distributed to them. If this data cannot be synchronized easily, a significant part of the promise of these devices will be lost. Although wireless connectivity is now being integrated into many types of product, it will still be a long time (if ever) before a mobile network connection becomes truly ubiquitous. Until then, data synchronization will be an important technology for ensuring that device functionality is not lost during times when the network connection is not available.

There are a number of companies selling data synchronization products today, offering a range of functionality on different platforms. Most synchronization products assume that the mobile device in question is connected to a PC, although the capability of synchronizing over the air is starting to become available. The problem is that all of these products use proprietary protocols, and so cannot interoperate. This can mean that it's very difficult to use the same dataset on different brands of device, or to allow users to choose *where* they would like their data to be synchronized. Different products also have diverse limitations on what data types they will support, and on whether they use open or proprietary data representation standards. The worst-case scenario is that we will see "islands of information" developing, where the boundaries of these islands are defined by proprietary synchronization protocols.

SyncML intends to prevent this from happening by being an open, freely licensable specification for data synchronization. The SyncML initiative was founded in February 2000 by Ericsson, IBM, Lotus, Motorola, Nokia, Palm Inc., Psion, and Starfish Software. As of June 2000, over 260 other companies have become supporters of the initiative. In order to overcome the issues outlined above, the founders decided to:

- Form an open industry initiative
- Generate a specification for universal data synchronization
- Facilitate data synchronization between any device and application, over any network
- Aim for interoperability between mobile devices and networked applications

SyncML is still a work in progress, although at the first SyncML Supporter Summit on 23rd June 2000, a number of demonstrators showed simple calendar synchronization using SyncML over a wireless connection. Psion, Nokia and Palm showed synchronization using HTTP over a dialup PPP connection, and Ericsson showed SyncML working over a WAP connection.

Since this is a public forum, and the SyncML specification is not yet publicly released, I am somewhat limited in the amount of detail I can describe here, but I will do my best to give a good overview of the new specification. If you are interested in finding out more and becoming involved in the evolution of the specification, details on how to join the SyncML initiative can be found at the end of this document.

Features of SyncML

SyncML has a number of properties that make it an attractive solution:

- Data type independence: it can synchronize more than just PIM data. Even custom-built applications can be synchronized with SyncML.
- Network independence: almost any transport can be used to transfer SyncML packages.
- Self-containment: each SyncML package is self-contained, and can be carried separately.
- SyncML is based on XML technology, which is well known and widely adopted.
- SyncML addresses the resource limitations of mobile devices, but without limiting itself to a "lowest common denominator" solution.

The SyncML initiative has no desire to "reinvent the wheel", so it was decided from the start that we would make use of existing open standards (especially for object types) wherever possible. This means, for example, that we will be reusing the work that has been done on vCard and vCalendar for contact and appointment types.

It should also be noted that the SyncML initiative is not a standards body. SyncML is in a pre-standards phase, and the initiative is dedicated to getting SyncML to a stage where there is sufficient momentum and market acceptance of the specification to justify its adoption by an international standards body. In other words, the initiative would like to put itself out of business, and the more successful SyncML is, the sooner this will happen.

SyncML is a specification, not a product. We are not trying to make the synchronization software vendors redundant; we are simply trying to establish a basis for true interoperability. POP3 significantly enhanced users' experience of e-mail by allowing different e-mail clients and servers to work together. By doing this it allowed greater innovation in e-mail clients and pushed that innovation to a higher level, where the changes made a much greater difference to the overall user experience and usability of products. SyncML aims to do much the same thing for data synchronization.

Components of SyncML

SyncML is composed of a number of components:

- SyncML Representation Protocol
- SyncML Synchronization Protocol
- SyncML Transport Bindings
- SyncML Object Representations

The aim of each component is to establish a specification for a particular function, in order to aid the overall interoperability of SyncML-conformant implementations.

SyncML Representation Protocol

SyncML defines synchronization as an exchange of **packages**, where a package consists of one or more **messages**. The SyncML representation protocol, which is defined using XML, defines the structure of such a SyncML message.

There are three DTDs (Document Type Definitions) defined as part of the representation protocol. The first defines the high-level representation of SyncML messages and commands, the second defines a format for meta-information, and

the third a format for information about the device and its capabilities (such as object type support and free memory). The namespace capabilities of XML are used to access elements from the second and third DTDs.

A SyncML message consists of a header and a body. The header holds session, routing and security information, and the body is a list of commands. The commands allow, among other things, the communication of any additions, deletions and updates necessary to bring two datasets into equivalence. For more complex operations, the subset of a database to be synchronized can be identified by a search operation, or multiple operations can be combined into a single operation with all-or-nothing semantics.

To allow for the fact that wireless connections have lower bandwidth than fixed connections, SyncML has made use of the binary XML coding developed by the WAP Forum. This binary coding reduces the amount of data that has to be transferred, and is also a good example of how the initiative prefers to reuse existing standards where possible. Plain text rendition of XML can still be used if required — this can be very useful while debugging!

SyncML Synchronization Protocol

The synchronization protocol defines the sequence of packages to be exchanged in order to perform synchronization. Both one- and two-way synchronization are supported. One-way synchronization is important, because it is then known that either only one copy of the data set is changing, or a user wants to save time by synchronizing their own changes without getting any changes back from the server.

The protocol defines two roles: client and server. Most of the time, synchronization is initiated by the client, although there is provision for the server to do so too. The server has the controlling role in terms of performing the synchronization analysis and having the first opportunity to detect and (potentially) resolve conflicting changes.

> *Since "client" and "server" refer to roles within the protocol, it is not necessarily true that the server role is performed by a large computer in a dark room! It is possible that both client and server roles could be performed by mobile devices. The important point about the server role is that it's where the synchronization analysis is done.*

As stated above, the client normally begins the synchronization process, although there is provision for the server to initiate where the underlying transport allows for unsolicited communication from server to client. It will also be possible to mix transport types, so that (for example) a synchronization server could send an SMS message (a "synchronization alert") to the device indicating that now would be a good time to synchronize. This will allow so-called "drip-feed" or continuous synchronization, meaning that the two datasets are equivalent within a very short time of any change.

The synchronization protocol contains safeguards to allow the detection of the loss of synchronization context. This would typically occur if the mobile device has been reset, and is resolved by performing a slow synchronization (in which *all* data in the dataset is exchanged, not just the changes since the last synchronization).

SyncML Transport Bindings

Although SyncML is transport independent, interoperability requires that there is a specification of how any particular transport is to be used to exchange SyncML messages. This ensures that there won't be any problematical issues regarding the setting of options within the transport — addressing, and the like.

The three transport bindings being defined in SyncML 1.0 are HTTP, WSP, and OBEX. HTTP (Hypertext Transfer Protocol) is widely used, and has the benefit of being able to pass through the majority of firewalls. WSP is the Wireless Session Protocol, a part of WAP, and allows WAP phones to support SyncML. Finally, OBEX (Object Exchange protocol) gives support for short-range connectivity. OBEX is currently defined over IrDA, Bluetooth, and USB.

SyncML Object Representations

The final piece that's required to enhance interoperability specifies the most commonly used data formats. The exact details of this are still being decided, but we will define the use at some level of vCard and vCalendar, possibly represented using XML. Any implementation that wants to be labeled as "SyncML-conformant" must use these representations for synchronization of contacts, appointments, to-dos, and the like. An additional work item right now is to define a representation for relational data.

The SyncML Reference Toolkit

The SyncML initiative is sponsoring the development of a C language toolkit that will be made open source once the specification is finalized. The toolkit encapsulates the encoding and decoding of SyncML messages, and is designed to be highly portable. It has a relatively thin layer that deals with memory management, file access and the like, but the vast majority of the code operates on memory buffers and so does not require much effort to port. The toolkit is being ported to Windows, Linux, EPOC and PalmOS. The exact open source license model to be used for the final code is not yet announced.

The Red release of the toolkit has already been made available to supporters of the SyncML initiative, and will be followed by Orange, Green and finally Gold releases during the course of this year. The Red release was used as the basis for the SyncML demonstrations shown at the Supporter Summit on 23rd June, and was ported to Windows NT, PalmOS and EPOC, as well as Nokia and Ericsson's own proprietary phone operating systems.

The reference toolkit has two main purposes. Firstly, it is intended to help potential implementers of SyncML by giving them code that is known to work on a variety of platforms and functions, in a way that is SyncML conformant. Secondly, it proves the specification by showing that it can be implemented in a sensible way. We made the decision early on that we could not release the SyncML 1.0 specification without having implemented it.

The server part of the demonstration code used at the Supporter Summit was specially written by the SyncML initiative. There is a simple synchronization engine, which has an API to the back-end database. So far, this API has been implemented for Lotus Notes and Microsoft Exchange. The architecture makes it relatively straightforward for the engine component to be replaced with different code, or for a different backend database to be supported. This was done deliberately so that the code gives plenty of assistance to supporter companies, allowing them to gain experience with SyncML as quickly as possible.

When will SyncML be Available?

Right now, we plan to release the SyncML 1.0 specification and reference toolkit in December 2000. No sponsors have made any specific product announcements as yet, but you should expect to see the first SyncML-conformant products being released shortly afterwards.

What does it mean to be SyncML-conformant?

Conformance is an issue that the SyncML initiative is discussing at present. Interoperability between different client and server implementations of SyncML is a key goal of the SyncML initiative, and we expect to make some announcements regarding a SyncML conformance scheme in the near future. There will be a number of interoperability testing events that will be open to supporters who are actively developing code.

How Can I Become Involved in SyncML?

Any company willing to sign the appropriate contract can become a supporter of the SyncML initiative. The contract is there primarily so that discussions about the specifications can take place within a legal framework that prevents the introduction of protected IPR (Intellectual Property Rights) into the specifications without prior notice and agreement. At the time of writing, there is no protected IPR within the SyncML specifications.

SyncML supporters get access to draft specifications for review purposes, and also get releases of the reference toolkit. There will be a second Supporter Summit in London in September, to which all supporters will be invited. This will consist of both business and technical presentations, along with live demonstrations of SyncML in action.

For more information on joining the SyncML initiative as a supporter, please see the SyncML web site at http://www.syncml.org.