

SyncML—Getting the mobile Internet in sync

Andreas Jönsson and Lars Novak

Today's business tools—desktop computer, laptop computer, personal digital assistant, mobile phone, and the applications that run on them—have made it much easier to access, use, and store a wealth of business data at the office or while on the move. The trouble is keeping calendars, address books, to-do lists, e-mail and other personal information management (PIM) data current across this variety of tools. The quest for seamless synchronization led to the development of several proprietary solutions to the problem. Unfortunately, most businesspeople do not tend to use applications and tools from a single vendor for all their daily communications and PIM needs.

Founded in February 2000, the SyncML initiative is a universal standard for synchronization across networks, devices, and applications from any vendor. The initiative is supported by Ericsson, IBM-Lotus, Matsushita (Panasonic), Motorola/Starfish, Nokia, Openwave and Symbian.

The SyncML initiative has worked quickly to develop and promote SyncML as an open industry specification for universal data synchronization. A live demonstration of the specification was conducted at the first SyncML Supporter Summit in June 2000 in Los Angeles. By December 2000, the specification was published as an industry standard (SyncML 1.0) and was launched at a public demonstration in London.

By the end of 2001, a new specification will have been introduced—SyncML 2.0. It will synchronize user data, enable mobile devices to be managed from servers, and promote new features, such as remote backup and restore. As mobile devices become more complex and are used to manage more complex tasks, these capabilities will be increasingly important.

The SyncML standard currently enjoys the support of nearly 700 companies, and additional companies are welcome to join and participate in the open initiative. This

year, the SyncML initiative was incorporated as a separate company, SyncML Initiative Ltd., which owns the specifications.

Synchronization—a real business need

An increasing amount of business is done by persons on the move. For example, according to management consultancy Booz Allen & Hamilton, 67% of professional workers in Europe are away from their desks or work area more than 20% of the time, which is to say that the need to keep appointment schedules, customer information, and task lists up to date while on the move has never been greater.

Synchronization enables changes to information stored on one device—for example, an address book entry on a mobile phone—to be automatically replicated either locally or remotely on all other devices that contain the same record, such as laptop PCs or the corporate network server.

Many business applications are facilitated through synchronization—for instance, maintaining up-to-date price lists and stock information for mobile sales forces. Changes made by a secretary to an appointment on the office network-based calendar application are automatically presented on the mobile phone calendars of the personnel involved. Likewise, changes made to the appointment by an individual on his or her mobile phone can automatically be replicated on the office system and then on the mobile phone calendars of all the other people involved in the meeting.

SyncML enables synchronization to be performed in a standardized way across applications, devices and networks—whether it is synchronizing e-mail in a pocket PC with that of a stand-alone PC via a Bluetooth or cable connection, or calendar entries on a mobile phone with those of the corporate network via the public mobile phone network.

With the any-device-to-any-device synchronization capabilities of SyncML, a mobile user who, say, receives an order via e-mail can then access the company stock inventory system on the same device to determine availability and delivery date.

Benefits that will drive the mobile Internet

SyncML will help drive the acceptance and use of the mobile Internet by helping to en-

BOX A, TERMS AND ABBREVIATIONS

API	Application program interface	RS-232	Recommended standard 232 (computer serial interface, IEEE)
DTD	Document type definition	SMTP	Simple mail transfer protocol
GPRS	General packet radio service	SyncML	Synchronization markup language
HTTP	Hypertext transfer protocol	TCP	Transmission control protocol
IMAP	Internet message access protocol	vCal	Calendar interoperability standard by the Internet Mail Consortium
IP	Internet protocol	vCard	Business card interoperability standard by the Internet Mail Consortium
IrDA	Infrared Data Association	WAP	Wireless application protocol
LAN	Local area network	WBXML	Wireless binary XML
MIME	Multipurpose Internet mail extension	WSP	Wireless session protocol
OBEX	Object exchange	XML	Extensible markup language
PC	Personal computer		
PDA	Personal digital assistant		
POP3	Post office protocol 3		

sure that the synchronization experience is smooth and simple for users, device manufacturers, network operators, service providers and application developers alike.

Users on the move will benefit from having access to a greater number of applications that are totally integrated into their corporate applications. Fast and simple synchronization will help users to keep their business lives well organized and under control. For example, SyncML means e-mail can be sent to and received from mobile devices without having to worry about reorganizing in- and out-boxes on the office PC. With only one synchronization protocol and procedure to consider, it will be easier and less costly to install, configure, and operate synchronizable applications across the organization.

Device manufacturers need only support one synchronization protocol that is interoperable with a broad range of applications, services, networks, and transmission technologies. This also frees manufacturers from having to allocate significant research and design effort to interoperability issues, and enables them to concentrate on the innovation and quality of their products.

Network operators and service providers will benefit from the interoperability of devices and the continuity of user experience—helping them to rapidly achieve a cost-effective critical mass for new services. Synchronization presents many new revenue opportunities through added service capabilities that will help create “sticky” mobile Internet applications, such as Web-based calendars and address books. What is more, network traffic will increase as more electronic devices support synchronization.

The universal interoperability that SyncML enables is also important for application developers, who will be able to design applications that can connect to a diverse range of device and networked data using only one synchronization protocol. This greatly simplifies and reduces the cost of the development process and facilitates updating or adding new applications.

Synchronization will become increasingly important with the advent of “always-on” mobile Internet and third-generation (3G) services.

Besides providing synchronization, SyncML might also be used as a highly efficient mobile Internet application platform. For instance, it is relatively easy to deliver services, such as stock updates, over SyncML.

All subscribers are potential synchronization service users, and soon most devices and applications will support SyncML.

Based on a study of the synchronization market, Gartner Group estimates that over the next few years the traffic generated by SyncML applications will exceed that generated from regular Web browsing. When machines begin communicating over the Internet using SyncML, network traffic will grow exponentially.

Open industry-standard technology

Where possible, SyncML makes use of existing Internet and Web technologies that are widely used and have been tested to ensure easy implementation and interoperability. To this end, SyncML uses the industry-standard extensible markup language (XML) for specifying the messages that synchronize devices and applications (using either plain text or the wireless binary XML, WBXML, binary encoding technique employed by WAP). This makes the protocol a future-proof synchronization platform.

SyncML is designed to be independent of data and transport type: it uses existing open standards for object types—such as vCal (calendar, to-do lists) and vCard (contacts) personal data formats—and can support arbitrary networked data.

To ensure interoperability, SyncML describes how common data formats are represented over the network. And to ensure extensibility, SyncML allows new data formats to be defined as the need arises.

SyncML can handle one- and two-way synchronization as well as client- and server-initiated synchronization. To ensure full interoperability, a range of transport protocols is supported. These include the WAP wireless session protocol (WSP), hypertext transfer protocol (HTTP) and OBEX (local connectivity using Bluetooth, IrDA and RS-232). SyncML can also be deployed over networks that use SMTP, POP3, IMAP, pure TCP/IP, and proprietary wireless communication protocols.

SyncML is not dependent on programming language. Nor does it assume that both ends of the synchronization process share a language environment. However, to facilitate rapid deployment of the protocol, SyncML Initiative provides a free reference toolkit written in C (the protocol can also be implemented in other languages).



Figure 1
Example of devices synchronizing data.

BOX B, ERICSSON AND SYNCML

Ericsson has been involved in the development and standardization of SyncML from the very beginning.

Ericsson was the first mobile phone vendor to demonstrate SyncML over the wireless application protocol (WAP)—initially, at the SyncML Supporter Summit in June 2000, and later, at the SyncML Supporter Summit in Dublin, as well as at the launch of SyncML 1.0 in London. At CeBIT 2001, Ericsson showed a complete SyncML solution, including the synchronization engine and clients.

In March 2001, Ericsson became the first mobile phone manufacturer to gain SyncML certification, for two GPRS phones, the R520 and T39 (Figure 2). Ericsson's synchronization engine, *Usync*, has also been certified.

Optimized for the mobile environment

SyncML has been optimized to meet the specific requirements of the mobile communications environment, taking into account the limited resources of mobile devices and networks and still offering 100% performance on other transport systems, such as cable systems or traditional local area networks (LAN).

SyncML has been designed to be robust enough to cater for the relatively high network latency (delay) in mobile networks, which increases as packet data techniques such as general packet radio service (GPRS) are introduced.

SyncML makes highly efficient use of the network—for example, only six packages need to be sent to achieve device synchronization. Minimizing request–response interactions in this way is important when synchronization is being performed over the air, where bandwidth is limited and packet costs are relatively high.

SyncML has also been designed to fit within the memory capacity of all common mobile devices on the market today, both in static code and run-time execution space. The data exchanged by the protocol is small (one or two bytes) and requires minimal code to transfer it to and from mobile devices. Exchanged data is generally binary-encoded (for example, using WBXML) to reduce the memory required to store synchronization

messages and to reduce the resources needed to process this data.

SyncML components

The main components of the SyncML specification are

- an XML-based representation specification;
- a synchronization protocol; and
- transport bindings for the synchronization protocol.

Representation specification

SyncML contains a set of well-defined messages that are conveyed between a client and server participating in a data-synchronization operation. The messages are represented as XML documents or as multipurpose Internet mail extension (MIME) entities. MIME, which is the Internet standard for identifying multipurpose message content, provides a useful mechanism for differentiating between separate content or document types.

SyncML supports a request–response data synchronization command structure as well as *blind push* commands.

The representation specification specifies an XML document type description (DTD), which allows the representation of all information required to perform synchronization, including data, metadata and commands. The synchronization specification specifies the SyncML messages that conform to the DTD, in order to allow a SyncML

Figure 2
The Ericsson R520 mobile phone (left) and T39 mobile phone (right).



client and server to exchange additions, deletions, updates and other status information.

Other DTDs define the representation of information on the device (such as memory capacity) and of various types of meta-information (such as security credentials).

Figure 3 shows the basic structure of a SyncML package—a conceptual container for synchronization messages—as defined by the representation protocol.

The SyncML message is an individual SyncML MIME entity; essentially a well-formed XML document.

The SyncML header contains routing, session, authentication and message information.

The SyncML body contains the various synchronization commands to be performed.

Synchronization protocol

The synchronization protocol supports one-way and two-way synchronization (client-to-server or server-to-client). Synchronization alerts generally come from the client, but there is also a provision for server alerts.

The synchronization protocol defines how the different messages in the SyncML DTD are to be exchanged according to a predefined schema.

In Figure 4, the mobile phone serves as the SyncML client and the server functions as the SyncML server. The SyncML client pushes SyncML request messages to the SyncML server. The server synchronizes the data within the SyncML messages with data stored in the server (including additions, updates, and deletions). The SyncML server then returns a response to the SyncML messages sent by the client. This simple example describes the roles of the devices in the SyncML specification:

- SyncML client—typically, this is a PC, mobile phone, or personal digital assistant (PDA). It contains a synchronization client agent and usually sends the SyncML messages (operations), possibly including payload data. It must also be able to receive responses from the SyncML server. In addition, it might, in some cases, be able to receive some SyncML messages as commands from the server side.
- SyncML server—typically, this is a networked server or a PC. It contains a synchronization server agent and synchronization engine, and usually receives the SyncML messages (operations), possibly including payload data from the SyncML client. The SyncML server must also be

able to send responses to the commands if needed. In addition, it might, in some cases, be able to send SyncML messages as commands to the client.

Transport binding

Although the SyncML specification defines transport bindings that specify how a particular transport is to be used for the exchange of messages and responses, the SyncML representation and synchronization protocols are transport-independent. Each SyncML package is completely self-contained, and could in principle be carried by any transport.

The initial bindings specified are HTTP, WSP and OBEX, but SyncML could also be implemented using e-mail, message queues or other methods. Because SyncML messages are self-contained, multiple transports can be used. Neither the server nor the client devices need to be aware of the network topology. So a short-range OBEX connection could be used for local connectivity, with the messages being passed on via HTTP to an Internet-hosted synchronization server.

SyncML framework

SyncML defines the format for synchronization data as well as a conceptual data-synchronization framework and data-synchronization protocol. The scope of the SyncML framework (Figure 5) consists of the SyncML format and a conceptual SyncML adapter and SyncML interface. This framework is useful for describing the particular system model associated with SyncML implementations.

The SyncML synchronization protocol, which is outside the SyncML framework, is essential for providing interoperable data synchronization. The SyncML data-synchronization protocol is defined by a companion SyncML specification (SyncProto). The application, *App A*, is a networked service that provides data synchronization with other applications—in this case, *App B*, on a networked device. The service and device are connected over a common network transport, such as HTTP.

App A uses a data-synchronization protocol, implemented as the sync engine process. The data-synchronization protocol is manifested on the network by client applications that access the sync server network resource. The sync server agent manages sync engine access to the network and communicates the data synchronization operations to and from the client application.

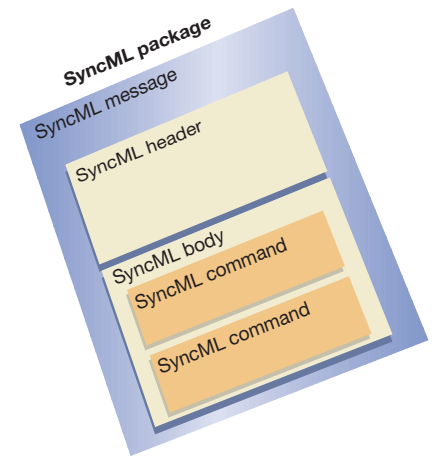
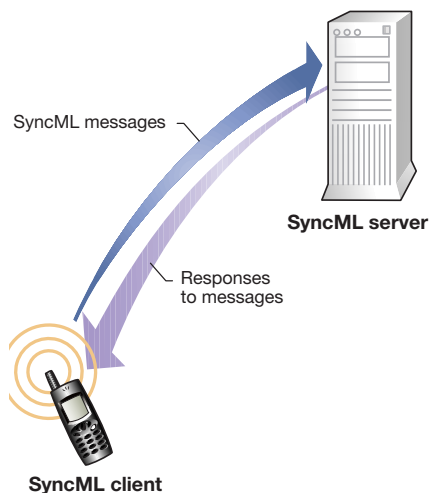


Figure 3 SyncML representation specification.

Figure 4 Example of synchronization example between a mobile phone and server.



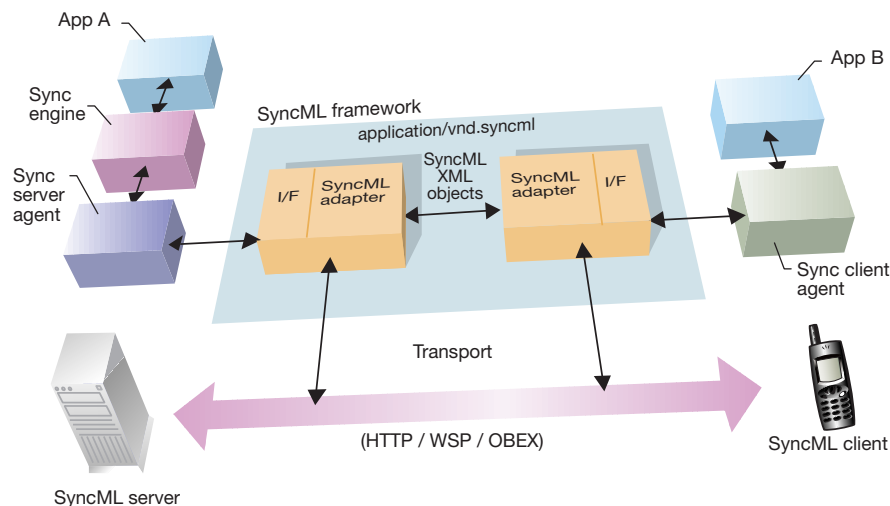


Figure 5
The SyncML framework.

BOX C, SYNCML COMMANDS

REQUEST COMMANDS

Command	Description
<i>Add</i>	Enables the originator to ask that a data element or collection of data elements supplied by the originator be added to a synchronization data collection that can be accessed by the recipient.
<i>Alert</i>	Enables the originator to notify the recipient.
<i>Atomic</i>	Indicates that a set of commands should be performed with all-or-nothing semantics. In addition, it indicates that a set of commands should be performed in the specified sequence.
<i>Copy</i>	Enables the originator to ask that a data element or collection of data elements accessible to the recipient be copied.
<i>Delete</i>	Enables the originator to ask that a data element or collection of data elements accessible to the recipient be deleted. A delete command can include a request for the archiving of the data. The deletion can either be a soft or hard delete.
<i>Exec</i>	Enables the originator to ask that the recipient invoke a named or supplied executable.
<i>Get</i>	Enables the originator to ask for a data element or collection of data elements from the recipient. A <i>Get</i> command can include the resetting of any meta-information that the recipient maintains about the data element or collection. This command must not be specified within a <i>Sync</i> command.
<i>Map</i>	Enables the originator to ask the recipient to update the identifier mapping between two data collections.
<i>Put</i>	Enables the original to put a data element or collection of data elements on the recipient device. This command must not be specified within a <i>Sync</i> command.
<i>Search</i>	Enables the originator to ask that the supplied query be executed against a data element or collection of data elements that can be accessed by the recipient.
<i>Sync</i>	Specifies that the included commands should be treated as part of an attempt to synchronize two data collections.
<i>Update</i>	Enables the originator to ask that a data element or collection of data elements accessible to the recipient be updated. An update can mean complete replacement of the data element. This command must only be specified within a <i>Sync</i> command.

RESPONSE COMMANDS

Command	Description
<i>Status</i>	Indicates the completion status of an operation or that an error occurred while processing a previous request.
<i>Results</i>	Is used to return the data results of either a <i>Get</i> or <i>Search</i> SyncML command.

The sync server agent performs these tasks by invoking functions in the SyncML I/F, which is the application program interface (API) to the SyncML adapter. The SyncML adapter is the conceptual process that the originator and recipient of SyncML-formatted objects use to communicate with each other. It is also the framework entity that interfaces with the network transport, and is responsible for creating and maintaining a network connection between *App A* and *App B*.

App B uses a sync client agent to access the network and its SyncML adapter, by invoking functions in the SyncML I/F.

Actual server and client implementations cannot be implemented in the discrete components identified by this conceptual framework. The SyncML specification does not attempt to impose this framework on implementers. However, the framework is useful for a discussion of the components that are necessary to implement a common data synchronization protocol.

SyncML package and messages

As touched on earlier, SyncML synchronization operations are conceptually bounded into a SyncML package. The SyncML package is simply a conceptual framework for one or more SyncML messages that are required to convey a set of data-synchronization commands. SyncML packages are not defined within the SyncML data representation DTD.

The SyncML message is an individual XML document that consists of

- a header—specified by the *SyncHdr* element type. The SyncML header specifies routing and versioning information on the SyncML message; and
- a body—specified by the *SyncBody* element type. The SyncML body is a container for one or more SyncML commands. The SyncML commands, which are specified by individual element types, serve as containers for other element types that describe the specifics of the SyncML command, including any synchronization data or meta-information.

SyncML commands

SyncML defines several “request” commands, some of which function as containers for other commands (Box C). For instance, the *Sync* command is used to establish equiv-

alence between two data sets and contains commands that specify changes to one of the data sets. Some commands can only be used either inside or outside of a *Sync* command. For instance, *Add* must only be used inside *Sync*, whereas *Put* must only be used outside of *Sync*. The two commands have similar semantics in terms of indicating an addition to the data set, but are defined separately to avoid overloading the same command with two different operations.

The SyncML commands do not fully define the semantics of the SyncML operation. For example, adding a document to an e-mail system database might have very different semantics from that of adding a transaction request to a queue. The type of data being synchronized determines the semantics of the SyncML operation. This means that it is possible for an originator to request an operation of a particular recipient that makes no sense to the recipient. In that case, the recipient must return an error response status code.

In SyncML, a data collection, *A*, has been successfully synchronized with data collection, *B*, if and only if a mapping, *M*, exists, such that for all identifiers, *I*:

A(I) exists, *B(M(I))* exists, and *B(M(I))* is equivalent to *A(I)*.

That is, if *A(I)* does not exist, then *B(M(I))* does not exist.

The exact definition of equivalence of data items is defined by the data synchronization model, and can be limited by the capabilities of the devices that hold the two data collections. For instance, a mobile device might not be able to store all the fields supported by a sophisticated server-based contact-management package. Nonetheless, the mobile and server data sets might still be considered as equivalent even though the mobile data set contains only a subset of the server data set.

Data synchronization using SyncML does not have to be symmetrical—one-way and two-way synchronization are supported. In this case, if *A-synchronized-with-B* is true, this does not imply that *B-synchronized-with-A* is also true.

SyncML data formats

SyncML also identifies a small set of data formats that provide a set of media types for exchanging common accepted information,

such as contacts, calendars, and messages. Support for these data formats is mandatory for conformance to the specification. In addition to these formats, SyncML allows for the identification of any other registered format. SyncML uses the MIME content type framework for identifying data formats, called MIME media types.

Capabilities exchange

SyncML supports capabilities exchange—that is, the SyncML client and server can determine which device, user, and application features the other device supports.

From the SyncML server perspective, the capabilities exchange is achieved through the *Get* command to retrieve the device information, user information, and application information documents from the SyncML client.

From the SyncML client perspective, the capabilities exchange is achieved by using the *Get* command to retrieve the corresponding documents from the SyncML server. These documents contain profile information on support for well-defined features.

The capabilities exchange can be used to establish or administer SyncML data synchronization services between a SyncML client and server.

Conclusion

SyncML provides an open industry specification for universal data synchronization. Launched at the end of 2000, it is now available to the industry free of charge, and SyncML-certified products are beginning to appear in the market. SyncML 2.0 (released in late 2001) also includes device management.

By providing a platform for interoperability between a whole range of devices, applications, and servers, SyncML is a true win-win solution for the entire telecommunications and IT industries, and will help drive the development and growth of the mobile Internet.

One of the founders of SyncML, Ericsson is also the leading manufacturer of products that support the standard—Ericsson has already announced mobile phones with SyncML as well as a SyncML server for corporate intranet access.