

Platform Independent Data Synchronization

Scenera Research LLC

Cary, North Carolina

www.sceneraresearch.com

Contact information, calendars, and all types of files on smart phones, PDAs, and computers are synchronized regularly by users and corporations. However, software programs that are currently available to perform synchronization have burdensome limitations and restrictions. Most are platform dependent, forcing users to buy their devices from the same vendor or limit them to devices that use the same operating system. Overseeing the scheduling, tracking, and documentation of synchronization is also difficult to manage with most existing systems.

The synchronization system we designed overcomes those limitations using presence services for real-time updates and an asynchronous publish/subscribe protocol for continuous Web-based operation. It can update any type or size of file that can be sent to and retrieved from a server and it can gather and store any type of data needed to log or document its operation. Synchronization can be automatically invoked when a device connects to the network so that all devices are always kept current. Most of all, any device that can hold two-way communications with a network server can be a synchronization client on this system, regardless of its manufacturer, vendor, or operating system.

Background

Computers and mobile devices allow users to collect and store data, from everyday contact and scheduling information to sophisticated dossiers and imagery. This information is typically stored in file systems, directory services, FTP mirrors, databases, and the like, and the information that is stored in one electronic device often must be kept identical to the information stored in others. In other words, all of the devices should be “in sync.” A common example of this is when a person owns a smart phone and a computer. The two devices have calendars, contacts, phone numbers, and files that are modified separately but must be kept identical on both. Another common example is corporations that issue numerous smart phones, laptops, and personal digital assistants (PDAs) to their employees and all must hold the same basic information.

Currently, electronic devices are synchronized through synchronization programs. However, most of those programs are platform dependent, being specific to a vendor, application, or

operating system. For example, ActiveSync™ is a free synchronization program from Microsoft® that automatically synchronizes Windows®-based computers and mobile devices. It uses a protocol named ActiveSync Exchange™, which is proprietary and requires device vendors to purchase a license from Microsoft® to achieve cross-compatibility. However, if a person with a Windows® computer uses a smart phone that is not compatible with ActiveSync™, the computer and phone cannot be synchronized unless another synchronization program is available that works across the platforms of both devices. Programs with the desired flexibility often have not been developed and data synchronization must be performed manually, which is a time-consuming and painstaking process that frequently furnishes error-ridden results.

With the ever-growing number and variety of different electronic devices in use worldwide and the growing capacity that they have for data handling and storage, we saw a need for a platform independent means of synchronizing electronic devices, and we designed such a system (1).

Synchronization Design

The platform independent, automated data synchronization system uses a principal-based publish/subscribe protocol and server such as those that currently support presence services. Publish/subscribe allows an entity known as a “subscriber” to subscribe to information provided by an entity known as a “publisher,” and presence services are just one type of application that publish/subscribe supports. As presence services demonstrate, the system operates in real time.

Publishers and subscribers are client devices that can connect to a network, such as personal computers (PCs), smart phones, and personal digital assistants (PDAs). Publishers post (publish) information to a Web address and subscribers gain access (subscribe) to the information by providing the same Web address, plus any IDs, passwords, or other information required for access.

The general configuration of this system is shown in Figure 1. It includes electronic client devices and a publish/subscribe server. The “data store client” shown in the diagram is the device that holds the new data that is to be shared with the other client devices. The data store client and the data recipient clients can be any types of electronic devices that can connect to one another through the server, and each of them must have at least one means of storing data, such as file systems, contact directories, calendars, relational databases, FTP mirrors, etc. This system provides platform independent data synchronization that can be used on any device that can communicate with a server.

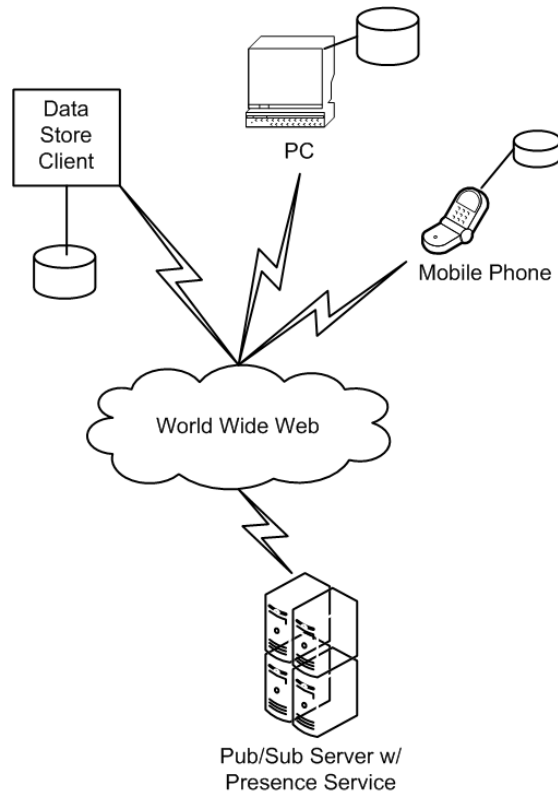


Figure 1. The general configuration of a platform independent data synchronization system that uses presence services with a publish/subscribe protocol.

Presence

Presence services convey the capability, availability, and willingness of a user or device to communicate with others in real time. Instant messaging is the application that is most well known for utilizing presence services, where it shows the availability of a “friend” or “buddy” through real-time indicators such as “online,” “in a meeting,” or “busy.”

Client devices that store and provide presence information are known as “presence clients” and applications that use information provided by presence are referred to as “presence-aware.” In device synchronization, all of the connected devices are capable of providing and receiving data updates, so all of them are capable of being presence clients and presence-aware.

Presence clients are classified as “presentities” and “watchers.” Presentities publish information to presence services for distribution, and presence services distribute published information to watchers, which subscribe to the published information.

The protocols most often used with presence are ones that meet the requirements and architecture described by the Internet Engineering Task Force (IETF) in Request for Comments

documents RFC 2778 and RFC2779. They are known as principal-based publish/subscribe protocols (2, 3).

Publish/Subscribe

Publish/subscribe protocols are simple and flexible. They require just three core commands to run operations on the Web, including PUBLISH, NOTIFY, and SUBSCRIBE:

- PUBLISH allows a presentity to provide or update information, such as device ID, status, and the data being updated or being used to update other devices.
- NOTIFY allows the server to provide information from a tuple to a watcher.
- SUBSCRIBE allows watchers to subscribe and unsubscribe to notifications.

Because these commands can be run on any devices that are configured to exchange information with a server, and because the publish/subscribe protocol requires so little memory, synchronization can run on virtually any type of device that can carry on two-way communications with a server.

The most common Web protocols today, such as HTTP and SMTP, are synchronous, meaning that their information and service providers must wait for requests before they can respond. Publish/subscribe protocols are asynchronous, meaning that they do not need to wait for requests. This allows them to provide publish/subscribe clients such as presence clients with new information as soon as it changes and in real time.

The presence publish/subscribe protocols are called “principal-based” because they publish information for identified users to tuples identified with the respective users, known as “principals.” Principals can be people, software programs, computers, mobile devices, or any other resource that can communicate with a server. The presentities and watchers are agents for principals. As described earlier, presentity clients are the publishers that provide the information to be stored and distributed through the server and watchers are the subscribers that receive published information from the server. With the synchronization system, the devices that are synchronized can function as presentities or watchers, depending on whether they are providing data to update other devices, or they are being updated by other devices.

A server can store information any way it wants and the synchronization system exchanges and represents the stored data in “tuples,” which are data objects with a schema defining conformance requirements for the data objects. In the synchronization system the basic components required for operation are a device’s ID, its current status (unavailable vs. available to receive a tuple), and the information to be shared. The size of a tuple is limited only by resource constraints, so the amount of data that can be shared can vary in size from very small to quite large.

Device Synchronization

During synchronization the devices to be kept in sync with one another are presence-aware and maintain links between one another that are like those between buddies in instant messaging. Also, each device is configured to publish its presence information through the presence service and each device that is online and subscribed to the service can receive published information as soon as it is available.

The presence information on each device includes a status value that indicates when its data has changed. When the status indicates that a change has occurred since the last update, a message with the status value and update is published to the publish/subscribe server. The server then sends a message with the new information to the subscribed devices, and those devices receive it with the status value. If the status value received by the devices indicates that any of them has not already received the update from that publishing client, then the synchronization starts. Once the data is synchronized, the status is reset to a value that indicates that synchronization with the publishing client has occurred and a message with the updated status value is sent to the server.

With this process, the devices that receive information are those that have a presence status value that indicates they are online and available. The general workflow of this process is shown in Figure 2.

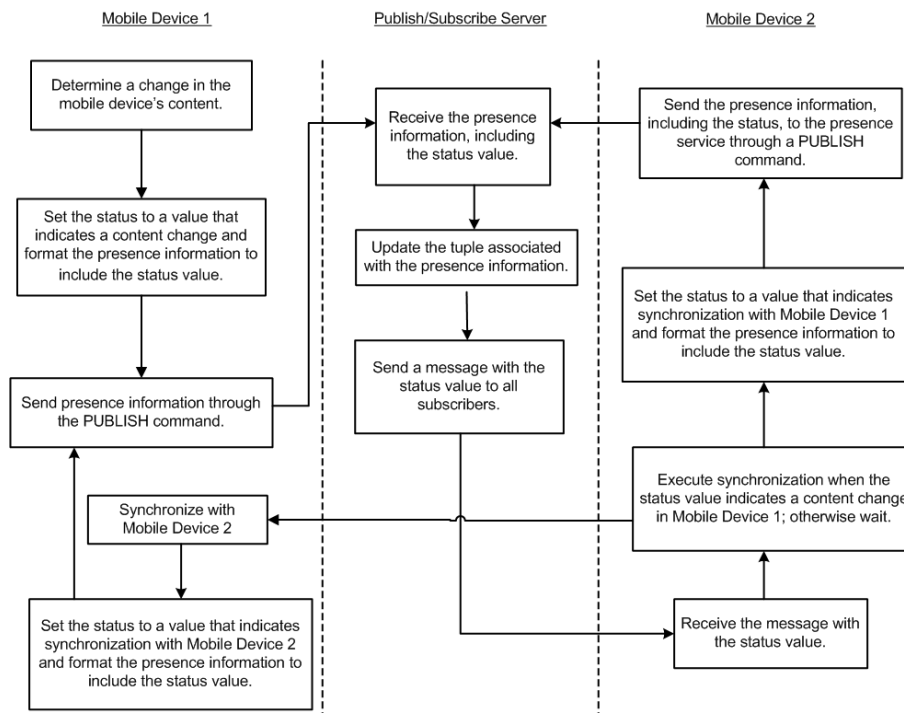


Figure 2. The general workflow during data synchronization between two mobile devices.

Because the publish/subscribe protocol is asynchronous, the device with changed data can have its status message distributed to any and all relevant receivers, simultaneously and in real time. Any subscribed device equipped with the precise tools to receive that message can be synchronized.

In instant messaging the real-time status that is provided, such as “available” or “unavailable,” is used passively between devices. This supports communication action only between the humans who view it. However, with data synchronization the status is designed to trigger communication actively between electronic devices. Together publish/subscribe and presence services allow client devices to check the current status of other devices, determine which devices are available to participate in a communication session, and transmit information updates to the devices that are online. Furthermore, the published tuple information is transmitted through messages that can be read simultaneously by any number of subscribers, and subscribers can receive a publisher’s posted messages as long as they remain subscribed to the service.

For example, if a telephone number on a smart phone has been updated and its owner has a laptop computer with a contacts list that must be kept in sync with the contacts list on the smart phone, the phone publishes a message to the server telling it that it has updated information. If the owner’s laptop is logged on the subscription server’s Web site, the synchronization starts.

If the owner’s laptop is not logged on the subscription server’s Web site, synchronization does not occur. As we designed the system, old data updates that have come and gone are irrelevant. The system does not queue information updates; it is designed to always synchronize data between all clients based on current data whenever they log on.

Synchronization without Queuing

With this synchronization process, all devices in the system can act as both publishers and subscribers, so when a change in the information of any device is indicated, it sends the updates to all online devices. However, because publish/subscribe services do not queue information the way that e-mail is queued when a user is not logged on, when an offline device logs on or subscribes to the publish/subscribe service, only the information in the currently published message is received. While this may imply that the subscriber device does not receive updates that circulated while it was offline or unsubscribed, in fact the subscriber is synchronized.

The tuple stores all updates and the status value, allowing all subscribed devices to recognize any updates not yet received. The current status for each device is kept on the publish/subscribe server, which keeps track of the status of each device. When a device that has been offline is logged on, the server receives its online status value and then sends the

device the tuple with all current information updates. During the updates and after all updates are made, the status value is set to a value that indicates where the device is with respect to the state of its data.

This system is more efficient than queuing because only current updates are received. If an entity, such as a document file or telephone number, has updated multiple times while a device is offline, the device does not go through all versions of each update. Only the most recent version is kept in the tuple and received by the subscribing device when it logs on.

This described system constantly circulates the status values. However, in systems with numerous mobile devices that are frequently connected to and disconnected from the server it may be better to manage synchronization from one point on the system, such as a database, rather than by circulating status values and data through all connected devices. This single-point design uses a synchronization manager that receives a notification message with its status value from each device. It stores the change information in a database and keeps track of the status of each client device. When it sees a client that is out of sync, it instructs it to use the change information to bring it in sync with the others in the group.

Synchronizations can be scheduled automatically using presence and publish/subscribe, but in a system with this design they do not need to be. All devices will be in sync when they are connected to the network.

The Benefits of Presence Synchronization

Using presence services to facilitate data synchronization provides many benefits. Foremost, the publish/subscribe communication protocol provides a common, platform independent command set that allows an electronic device to publish its presence information and to subscribe to other clients' presence information without regard to platform specific factors, such as vendor, application, or operating system. This allows different devices with their data store systems to exchange synchronization information with each other using a standard protocol.

Data updates also can be organized and managed by connecting the system to a database. Using this tool would allow an organization to make updates that can be distributed to all devices connected to its synchronization operation. Using presence services with publish/subscribe protocols, multiple connected electronic devices can receive the published information at the same time and in real time.

By including additional data components in a tuple, this system can provide details about the synchronization, such as data needed to verify updates and information needed for documentation. This can include an element that captures the device client's operational status, such as offline, active, or backing up, or a timestamp that records when the last

synchronization operation occurred. For further documentation, sub-tuples can be used to record versions of data updated, and when a change includes the removal of data it can be used to record the data deleted. Virtually any type of information that is available on the devices or the server can be recorded or backed up using tuples.

With the wide range of available data, updates can be prioritized by date and time, rank, affiliation, type of device, user, or any other data that can be gathered in a tuple.

This is a flexible, adaptable system that can continuously synchronize a wide variety of data between any number of devices that can access and log onto its networked Web site.

Acknowledgement

Scenera Research LLC thanks Julie Tomlinson for her writing and editing.

References

1. R.P. Morris. 2008. Method and system for synchronizing data using a presence service. U.S. Patent Application No. 20080027996. 24pp.
2. M. Day, J. Rosenberg, and H. Sugano. 2000. A model for presence and instant messaging. Internet Engineering Task Force, RFC 2778: <http://www.ietf.org/rfc/rfc2778.txt>.
3. M. Day, S. Aggarwal, G. Mohr, and J. Vincent. 2000. Instant messaging/presence protocol requirements. Internet Engineering Task Force, RFC 2779: <http://www.ietf.org/rfc/rfc2779.txt>.

Contact

Contact Scenera Research at info@sceneraresearch.com.