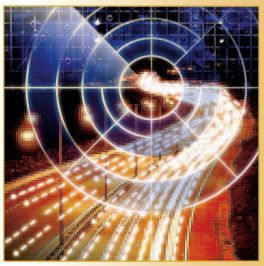
Editor: Jim Whitehead, ejw@cs.ucsc.edu



Open Calendar Sharing and Scheduling with CalDAV

Building on a decade of work on calendar standards, the CalDAV protocol promises to unlock the potential of widespread calendar interoperability. It permits calendar sharing over the Web and reduces the coordination cost of scheduling meetings across and within organizational boundaries. The protocol extends the Web Distributed Authoring and Versioning (WebDAV) protocol — itself, a simple extension of HTTP — to provide services for calendar maintenance, queries, event scheduling, and security.

Lisa Dusseault
Open Source Applications Foundation

Jim Whitehead
University of California, Santa Cruz

Consider for a moment the effort it takes to schedule a meeting that includes people from multiple organizations. Currently, doing so involves phone calls and multiple rounds of email, usually coordinated by one person, to build consensus for a time that works for most people; the coordinator must also delicately handle cases in which the chosen time excludes one or more participants.

For meetings in which everyone comes from the same organization, participants can solve scheduling issues largely via calendar sharing and scheduling software. To schedule a meeting with Oracle Calendar or Microsoft Exchange/Outlook, for example, you log in to a calendar server and then issue search requests for available times among a group of meeting participants. Once the application finds a free time slot, it sends a meeting request and then handles acceptance or rejection replies. Organizations that adopt this technology can see dramatic benefits as meeting-coordination time drops from tens to single-digit minutes – as long as everybody uses client software that supports the chosen enterprise application.

Unfortunately, these applications work only within existing organizations. Consider what happens when Jim, who works outside the organization, wants to schedule a meeting with Larry and his coworkers, who are inside. Because he doesn't have an account on the internal calendar system, Jim must send an email to Larry, who then takes on the coordination task to find meeting slots that work for his coworkers. Larry must then negotiate with Jim for a time slot that works for members of Jim's organization. Even if Larry and Jim both had access to calendar systems for their respective organizations, the process would remain tedious and prone to coordination breakdowns.

To efficiently schedule meetings, users need an interoperability protocol that lets

 1089-7801/05/$20.00 © 2005 IEEE Published by the IEEE Computer Society

a diverse range of calendar clients and servers communicate over the Internet. With such a protocol, we could use existing tools to schedule cross-organization meetings as efficiently as within-organization meetings. This protocol would also make it easier to support calendar access and scheduling functionality on a broad range of devices by providing an open, consistent, standardized way to retrieve calendar data.

History of Calendar Interoperability

Protocol developers have long recognized the need for a calendar access and scheduling protocol. A July 1996 press release from Netscape Communications heralded the formation of a working group dedicated to developing standards for calendaring and scheduling on the Internet.[1] That group became the IETF's Calendaring and Scheduling (CalSch) working group, which operated from October 1996 through September 2004. CalSch initially divided its work into three main lines of development:

- a data model and textual representation for calendar events (which generated the iCalendar specification),
- the transport of calendar information via email and LDAP (which resulted in the iCalendar Message-Based Interoperability Protocol [iMIP]), and
- a general-purpose specification for calendar access and scheduling (which became the Calendar Access Protocol [CAP]).

A key problem in developing interoperable calendar applications is determining a standard way to represent calendar items, including those that repeat over time (a meeting held every Monday at 11 a.m., for instance). CalSch built on earlier work by the Versit consortium, which developed an initial calendaring and scheduling specification called vCalendar (www.imc.org/pdi/). After two years of refining that work, CalSch produced RFC 2445,[2] which is now in widespread use. The iCalendar data format it defines shares vCalendar's non-XML format for representing attribute-value pairs.

CalSch also developed the iCalendar Transport-Independent Interoperability Protocol (iTIP) for calendar retrieval and scheduling operations.[3] This document described conceptually how to perform calendar-related operations, but it didn't provide a concrete, on-the-wire protocol. The iMIP specification,[4] which describes how to perform iTIP operations via email, has seen some success, including several vendor implementations and some cross-vendor interoperability. However, iMIP isn't commonly used today by typical calendaring or email applications.

CAP provides services that calendar applications can use to access calendar items from remote servers, search for open time periods in another person's calendar (known as free/busy queries), and schedule meetings. (See www.calsch.org/ietf/drafts.html for a complete list of CalSch's documents, drafts, and issues.) In versions 00 through 05 of CAP (released between August 1999 and July 2001), CalSch developed an entirely new protocol that was distinct from all existing application-layer protocols, although it borrowed somewhat from the Post-Office Protocol (POP) for its interaction style. In versions 06 through 11 (November 2001 through July 2003), the working group used the Blocks Extensible Exchange Protocol (BEEP) for its marshalling syntax and messaging behavior.[5] CalSch made no further progress on CAP, and the IETF closed the working group in September 2004. After four years of development, CAP was dead.

As CAP development was slowly progressing, several implementers were routing around the working group to release functional Internet calendars. In 2002, Apple Computer released its iCal personal calendar application, which supports Internet-based calendar sharing. With iCal, a user can publish a calendar to a server running Web Distributed Authoring and Versioning (WebDAV),[6] from which anyone else can view and download events. Although Apple designed the application to integrate with its WebDAV-based .Mac service, iCal works with any WebDAV server. A few open-source clients adopted iCal-over-WebDAV as the de facto first open calendaring standard.

In WebDAV,[6] Apple made an interesting choice. The protocol extends HTTP to include overwrite prevention (locking), namespace operations (list collection, move, copy, create collection), and metadata (properties). Combined with HTTP's capabilities for reading, writing, and deleting Web resources, WebDAV provides all the features necessary for remote publishing and sharing of calendars. Given that Apple was already using WebDAV for access to its .Mac Internet disk service, it was able to piggyback calendar sharing on top of the existing WebDAV server infrastructure — a far more attractive option than implementing and fielding server infrastructure for a new protocol

such as CAP. Apple showed, in a very public way, that calendars could be treated like any other Web resource accessible via HTTP.

Although Apple's iCal provides useful capabilities for individuals to publish and share their calendars, it does have drawbacks for corporate use. It isn't easy to search for free/busy times across a large set of people or to find other people's calendars. Due to synchronization issues, iCal also makes it very tricky to have someone else manage your calendar for you. These problems largely stem from protocol-level shortcomings. Although native WebDAV easily supports individual publishing and sharing, it doesn't provide any support for calendar locating, searching, or workflow scheduling. Yet, Apple's success with iCal-over-WebDAV raised the question of whether specialized calendar support could be added to WebDAV, rather than requiring a new protocol like CAP. Ideally, we would like a calendar access protocol to support standard within-organization meeting scheduling, as well as collaborative calendar sharing such as a family might use (Figure 1). Additionally, we want to make it much easier to schedule a meeting between participants from multiple organizations (Figure 2). Supporting these scenarios is the motivation for the Calendaring and Scheduling Extensions to WebDAV (CalDAV) protocol.[7]

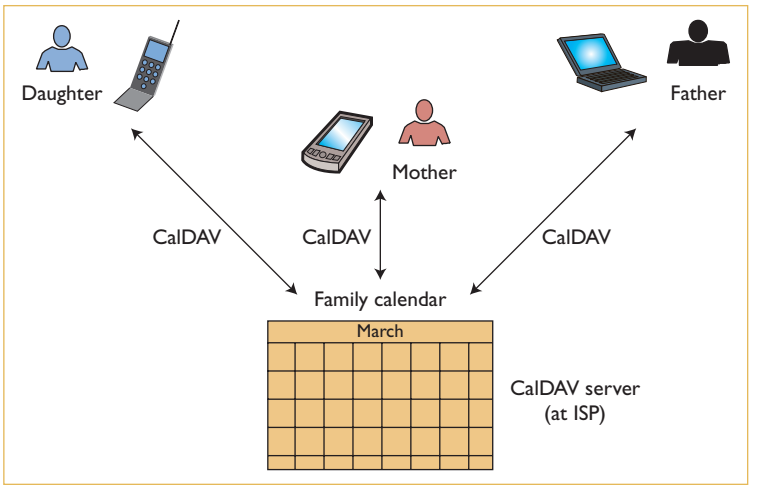


Figure 1. Collaborative editing scenario with a shared calendar. This family uses a shared calendar maintained on a CalDAV server run by their ISP. The father uses a laptop computer at work and home to update the calendar; the mother uses a PDA with wireless access. Their daughter keeps in synch with the rest of the family by viewing and updating the calendar from her cell phone.

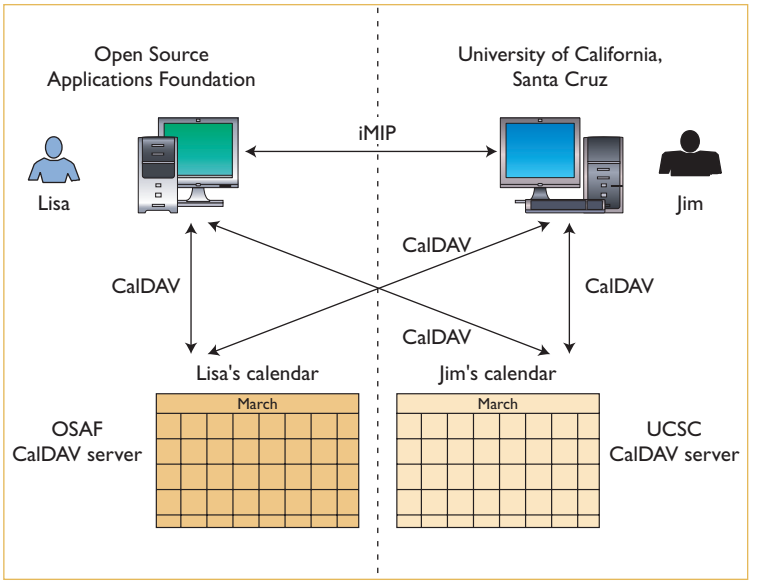


Figure 2. Scheduling across an organizational boundary. Lisa and Jim work for two separate organizations and need to schedule a meeting. Using her CalDAV calendar client, Lisa first searches their calendars for free/busy times on the day she'd like to meet. She then sends a meeting invitation, which appears in Jim's event inbox. Jim accepts the meeting and updates the calendars to show this.

CalDAV in a Nutshell

The base CalDAV protocol provides three main features:

- *Calendar maintenance*. Users can create multiple personal calendars (one each for work, conference time slots, home, and so on) via a new `mkcalendar` method.
- *Calendar queries*. People can search other people's calendars for free/busy times, or they can discover who is participating in a given meeting. Calendar applications can use queries to discover when to-do list items are due, and a flexible new `report` type supports a wide range of calendar queries.
- *Calendar security*. Users can control how much of their calendars are visible to others, and who has permission to change them, by using CalDAV extensions to the WebDAV access control protocol.

Using the same framework, a separate draft will build on the core CalDAV specification to provide optional scheduling workflow functionality. This will let people using calendar applications make and reply to meeting invitations with a new `schedule` method along with inbox and outbox collections, and *invitation fanout* (replication) rules. Meeting participants can be from a single

```
>> Request <<

GET /bernard/calendar/inbox/mtg456.ics HTTP/1.1
Host: cal.example.com

>> Response <<

HTTP/1.1 200 OK
Date: Thu, 02 Sep 2004 17:05:23 GMT
Content-Type: text/calendar
Content-Length: xxxx

BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//Example Corp.//CalDAV Server//EN
BEGIN:VEVENT
DTSTAMP:20040901T200200Z
DTSTART:20040902T130000Z
DTEND:20040902T140000Z
SUMMARY:CalDAV draft review
UID:34222-232@example.com
ATTENDEE;PARTSTAT=ACCEPTED;ROLE=CHAIR;CUTYPE=
      INDIVIDUAL;CN=Lisa
Dusseault:http://cal.example.com/lisa/inbox/

ATTENDEE;PARTSTAT=NEEDS-ACTION;ROLE=REQ-
PARTICIPANT;CUTYPE=INDIVIDUAL;
CN=Bernard
Desruisseaux:http://cal.example.com/bernard/inbox/

ATTENDEE;PARTSTAT=NEEDS-ACTION;ROLE=REQ-
PARTICIPANT;CUTYPE=INDIVIDUAL;
CN=Cyrus Daboo:http://cal.example.com/cyrus/inbox/
 END:VEVENT
 END:VCALENDAR
```

Figure 3. Using HTTP `get` for a calendar event. The request shows an HTTP `get` submitted to a CalDAV server to retrieve a calendar event. The response shows an iCalendar event in the HTTP response body with a MIME type of text/calendar. The meeting is titled, "CalDAV draft review," has three participants, and occurs on 2 September 2004, from 13:00 to 14:00 hours.

organization or span many. A person sends a meeting invitation by instructing the calendar application to invoke `schedule` to place the meeting in their outbox, thereby causing invitations to be placed in other attendee inboxes.

Modeling Calendaring Objects as HTTP Resources

HTTP resources have URLs and respond to `get` requests. Calendar applications can also delete and write them as whole resources by using `delete` and `put`. Although this is more a description than a definition, it's useful to help decide what to represent as resources in a given application. What size object should have its own address? What size resource is most useful to download or upload? What size resource is most useful to create or destroy?

Within the iCalendar data format used by CalDAV, a calendar is composed of events, each representing some activity a person or group will attend — meetings, appointments, performances, trips, or essentially anything with a start and end time.

In CalDAV, every event is represented as a resource, and resource contents are represented on the wire in the iCalendar text format.[2] Given that user operations often involve single calendar events, being able to provide a URL as an address for each event and to create, delete, or overwrite them is very convenient.

It might also be useful to have URLs and operations for an event's individual properties (such as its location, or start and end times), but that level of granularity is not required for reasonable performance. Although some use cases involve changing only an event's location, it's possible to do so within reasonable transmission and processing times by rewriting the whole event. There aren't strong use cases for creating a resource that represents an event location without an event that has that location. The working group made a similar choice with the iCalendar standard in defining event properties (except for `timezone`) only within the context of an actual event. To keep things simple, CalDAV doesn't define URLs for event properties.

Given that CalDAV represents calendar events as HTTP resources, the protocol must decide which body and MIME type to assign these resources; this choice determines what the result of a `get` looks like. The working group opted to use the iCalendar standard to represent all event data within the HTTP resource's body. This makes use of existing work on the iCalendar standard, software libraries that implement iCalendar, and iCalendar interoperability experience. In fact, Web servers can already store files with the MIME type "text/calendar," and it's not unusual for existing Web browsers to dispatch downloaded iCalendar files to calendar applications to handle appropriately. Figure 3 shows an HTTP `get` request for an iCalendar-formatted event stored on a CalDAV server.

Recurring Events

A calendar repository needs a clear model of how to deal with recurring events. Imagine the confusion if one client represented recurrences as a property but another client couldn't understand the syntax and saw only one instance of the event. The situation gets even more complex when we start modifying recurrences. The specification thus needs to be clear on whether a client can modify an existing resource or must create a new one to create a new instance of a recurring event.

The choice of iCalendar provides guidance here because it almost always represents a recurring event as a single component with a set of recurrence dates or patterns. Following this pattern allows CalDAV to represent even infinitely recurring events in a noninfinite space — a definite plus for synchronizing an entire calendar.

Using HTTP Features on Calendars

With iCalendar events mapped to resources, we can use base HTTP to quickly solve an important set of use cases for CalDAV:

- To download an event, a calendar application uses `get` and interprets the body of the response as an iCalendar file.
- To create a new event, the application uses `put` (and an unmapped URL) and sends an iCalendar file in the request body.
- To overwrite an existing event or to change its location, time, or attendees, the calendar application sends a `put` request to an existing event URL with the new values for the event in the iCalendar-formatted request body.
- To delete an existing event, the application uses `delete` on the event URL.
- To check whether an event has changed since last downloaded, the application uses the HTTP `ETag`.

Although HTTP supports many useful calendar operations on its own, it has its limits. HTTP has no capability for listing all of a person's calendar events or preventing collaborators from overwriting a shared calendar. The WebDAV protocol does support these functions; hence, CalDAV builds on both HTTP and WebDAV.

Using WebDAV Features on Calendars

The functionality WebDAV provides on top of HTTP is also important to CalDAV. Events aren't completely unassociated with each other; instead, a CalDAV server collects them into calendars that roughly represent the time commitments for a given person, object, or place. Sometimes it's useful to see a single event, but other times we need the contents of a whole calendar or some time-slice thereof. For reasonable synchronization, it also helps to see which events have been added or removed since the last calendar synchronization.

In these use cases, CalDAV uses the WebDAV *collection* resource to model calendars. A WebDAV collection provides a way to list its resources in a clear, flexible, and machine-parsable (XML) way. And because we've already defined events as resources, WebDAV operations work on events and calendars in a straightforward way:

- To list all events in a calendar, use `propfind` on the calendar URL.
- To list all the calendars in a certain area of a repository, use `propfind` on the appropriate URL.
- To copy an event to another calendar or create a duplicate event within one calendar day, use `copy`.
- To rename an event or move it to another calendar, use `move`.
- To lock an event or to lock an entire calendar while making changes, use `lock/unlock`.
- To synchronize a calendar that has been downloaded before, use `propfind` and ask for the `getetag` WebDAV property, which provides the HTTP ETag value and shows new resources and changed and unchanged ETags, and omits deleted resources.

WebDAV also defines the `mkcol` method, which creates collections of resources, but CalDAV uses a different method because a calendar is a bit more than an ordinary collection, and the CalDAV client must signal that intention to the server.

New CalDAV Mechanisms

The native capabilities of HTTP and WebDAV support many useful calendar-retrieval and update scenarios. However, it is necessary to extend WebDAV's core capabilities to provide rich support for calendar queries for free/busy times, creating collections of events, and supporting meeting invitation workflows.

Querying Calendars

The major use case that the HTTP and WebDAV methods we've described so far can't handle is the ability to search a calendar. A generic search

mechanism that names properties and property values might seem attractive, but it doesn't work well when dealing with time ranges and recurrences, which are necessary complications of the event model defined in iCalendar. Alternatively, it is possible to download a copy of someone else's calendar and then perform free/busy queries over this local data. Calendars typically have many entries, and the time required to download a local copy is far greater than asking a server to perform the equivalent free/busy query.

When scheduling a meeting, the most common use case is to request all events in a given time range. To schedule a meeting with a busy coworker, for example, you might want to see her schedule for the next week, including all recurring meetings and any multiday meeting that overlaps with the next week. This isn't a complicated request, given a syntax that's tailored to time-range objects.

CalDAV solves this by borrowing the `report` syntax defined in RFC 3744[8] and defining a calendar-specific time-range report.[9] This report provides a way to compile information in a single request and response that would be prohibitively expensive to collect using only `get` and `propfind` requests. At the same time, this report solves the problem already noted of expanding recurring events and identifying which events occur in a given time range. If the client asks for all events with start times after 8 a.m. this morning and end times before midnight tonight, for example, it could miss recurrences. With a `report`, on the other hand, the server can perform the necessary time and recurrence calculations and provide the exact set of events that should show up in a given time period.

Identifying and Creating Calendars

The need to be able to identify calendars follows from the need to support special reports on them. To treat a collection as a calendar, the client has to both know that it is a calendar and support the calendaring functionality. The mechanism that WebDAV defines for this purpose is the `resourcetype` property, and CalDAV defines a new value for it: `CALDAV:calendar`.

The need for a special mechanism to create calendars then follows from the need to identify them. To ask servers to create calendars, clients need a method that indicates what kind of resource to create and what functionality to provide on it.

In CalDAV, the client simply sends a `mkcalendar` request to the server, which creates a calendar (if it can) in a user's calendar storage space on a CalDAV-compliant server. The server automatically assigns the correct `resourcetype` value, which enables other clients to detect that the new resource is a calendar that contains events.

Scheduling Meetings with Base CalDAV and iMIP

Scheduling meetings is already possible with the CalDAV features we've discussed (all of which are described in the core CalDAV specification). A client can use a report to view another user's free/busy time, and then use the iMIP standard to send an invitation over email. CalDAV increases the likelihood that the meeting invitation will avoid time conflicts.

Although this approach works for many applications, however, it's not sufficient for enterprise-level calendaring and scheduling. In existing systems (using proprietary protocols), the server generally does more to help schedule — in part, to improve the user experience, but also to help address email's shortcomings as an application transport.

These shortcomings arise from the history and practice of handling email and spam. Mail can be delivered through several mail-transfer agents (MTAs) before arriving at one or more mail user agents (MUAs), as the IETF calls mail-reading applications. This architecture works great for email, but it was never designed to deal with things like iCalendar invitations. Some of the problems it presents include:

- How does calendaring software know which MUA will receive invitations?
- How do MUAs know which application to send invitations to?
- Some email software bypasses these last two issues with an integrated calendar module, but what if users want to manage their calendars with a different application?
- What does the MUA on a PDA or cell phone do with an invitation if no calendar software is available?
- What do MUAs do with invitations that appear in inboxes?
- How do multiple MUAs — each of which will discover a given invitation in the user's email inbox — know which MUA should deal with the invitation and whether it can be deleted?
- What do MUAs do with invitations that have

Learning More about Calendaring

Information about Internet calendaring and scheduling is tricky to find because it is spread across multiple unconnected Web sites. Unfortunately, no books cover the iCalendar standards, but the following list highlights some of the best sources of information about iCalendar, CalDAV, and WebDAV.

- The CalDAV Resources site has links to the CalDAV mailing list, protocol drafts, and recent CalDAV news (http://ietf.webdav.org/caldav/).
- The IETF's Calendaring and Scheduling working group (www.calsch.org) maintains a list of calendaring RFCs and revision histories for the iCalendar, iCalendar Transport-Independent Interoperability Protocol (iTIP), iCalendar Message-Based Interoperability Protocol (iMIP), and Calendar Access Protocol (CAP) specifications. It also maintains a rich list of pointers to calendaring products, articles, and open-source projects.
- The Internet Society's Internet Report site houses the most recent CalDAV specification, along with its revision history (http://ietfreport.isoc.org/idref/draft-dusseault-caldav/).
- The Open Source Applications Foundation maintains the CalDAV mailing list, which is open to all (http://lists.osafoundation.org/mailman/listinfo/ietf-caldav).
- The Calendaring and Scheduling Consortium's CalDAV Technical Committee maintains a limited Web page about CalDAV efforts (www.calconnect.org/tc-caldav.html).
- Lisa Dusseault's book, *WebDAV: Next-Generation Collaborative Web Authoring* (Prentice-Hall, 2004), provides a detailed description of the WebDAV, Access Control, and DeltaV protocols.
- Julian Reschke's Greenbytes WebDAV page (www.greenbytes.de/tech/webdav/) tracks ongoing activity in the WebDAV protocol space including the CalDAV specification.
- Finally, last issue's Standards track article, "WebDAV: Versatile Collaboration Multiprotocol" (*IEEE Internet Computing*, Jan./Feb. 2005, pp. 66–74), provides an overview of recent activity in the IETF WebDAV working group.

already been moved to another folder, possibly by categorization rules or by another MUA?

Recent antispam measures have made meeting scheduling using iMIP even more difficult. MTAs often alter or block messages with unrecognized attachments, and MUAs could mark calendar invitations as spam.

Finally, to make scheduling workflows more effective, existing enterprise calendaring systems include mechanisms for receiving and recognizing invitations before they're even delivered to the calendar application. While on vacation, users often leave their computers off, so they aren't checking for, accepting, or publishing invitations on the server. In the meantime, coworkers might want to schedule meetings for as soon as vacationers return. When a calendar server can receive and handle invitations on a user's behalf, it can tentatively place such invitations on the user's published calendar. Although the meetings aren't yet accepted, coworkers can see which blocks of time might already be filled, so they can try to find unrequested times for other meetings.

Scheduling using CalDAV Directly

Given the advantages of scheduling via a calendar server rather than email, the CalDAV specification authors already have a separate draft in progress to define how to solve scheduling use cases with a CalDAV server's assistance. This work is still in greater flux than the base specification, partly because the challenges are a little more difficult.

With WebDAV, clients access stored resources, rather than receive them from servers. It assumes that resources are somewhat stable, so the model doesn't fit as well for applications in which messages are generally in transit. Thus, it's tempting to treat scheduling as a simple repository-access problem: to schedule with Jim, Lisa should create an event in his calendar, and vice versa.

The straightforward repository-access approach doesn't suit the way users handle events. Rather than letting everyone who is allowed to invite her to meetings simply put them on her schedule — implying confirmed attendance — Lisa wants to treat each new scheduling item as a request to be reviewed.

The authors of CalDAV's scheduling support are currently looking at *scheduling inbox collections* to solve that problem. Jim could put a request in Lisa's scheduling inbox, where it would be considered tentative until she moved it to the main calendar. This approach seems to work better with common scheduling workflows, but it still leaves many unresolved details. For example:

- If Jim can place events in Lisa's scheduling inbox, does that mean he can change events in that collection, including the one he created?

- When Lisa accepts a request, does the client put it on the calendar, or does the server handle that?
- Is there more than one scheduling inbox if Lisa has more than one calendar?
- What happens to a meeting request after Lisa accepts it?
- Can a user review the request history for a meeting, including updates as well as the original request?
- Can a user review the history of outgoing requests, including acceptances and requests that counter a suggested meeting time with an alternate?

The CalDAV draft authors and CalDAV mailing list participants are currently discussing how to resolve these issues.

CalDAV Status

After consuming significant committee time in debates over requirements, scope, and basic model assumptions, previous attempts at standardizing a protocol for calendar access proved inconclusive. Part of the difficulty is that many different models exist and work well for the applications that use them. Although it's possible in many software areas to define a single standard that encompasses most of the features in existing nonstandard applications, this hasn't proven feasible for calendaring applications. Instead of a committee approach, the dedicated work of a few like-minded individuals thus seems likely to make more progress at this point.

Because CalDAV is an individual submission to the IETF, the current work on it is in the hands of the draft authors:

- Cyrus Daboo, chief technology officer of Isamet (www.isamet.com),
- Bernard Desruisseaux, who works on the Oracle Calendar server (www.oracle.com/collabsuite/), and
- Lisa Dusseault, of the Open Source Application Foundation (www.osafoundation.org).

To streamline the standardization process, they plan to develop CalDAV quickly in a small but open group. They welcome input, either in person or through the open mailing list at http://lists.osafoundation.org/mailman/listinfo/ietf-caldav. Once they have a stable proposal with interoperable implementations, the authors plan to introduce CalDAV fully into the IETF standardization process.

Currently, the Calendaring and Scheduling Consortium (www.calconnect.org) is contributing to the requirements for CalDAV and supporting interoperability testing events. Although it isn't a standards organization, CalConnect is working to improve the interoperability of calendaring and scheduling applications as well as public awareness of standards in this area.

Isamet (client), Oracle (server), Mozilla (open-source client), and Slide (open-source server) have already begun developing CalDAV implementations. The first interoperability test event took place in January 2005, and demonstrated CalDAV interoperability among two servers and three clients.

Work continues on refining the CalDAV specification according to implementation experience, and those involved expect to make the first deployment of CalDAV capabilities available to users within the coming year. Once CalDAV demonstrates a track record of interoperability and the base of users working with CalDAV-aware clients and servers increases, we expect other calendar client applications and servers to adopt it — especially given that CalDAV is the only viable calendaring access protocol.

CalDAV will help millions of users schedule meetings within and across organizations using their PCs, personal information managers, and cell phones. It will remove much of the accidental complexity of gathering people together for a common purpose, freeing us for more creative and productive work. Moreover, because open standards beget open-source implementations, CalDAV promises to bring advanced calendar scheduling capabilities to families, small nonprofits, schools, and many others for whom current calendar technology is too complex and expensive.

References

1. "More than 20 Companies Join Netscape to Help Define Open Standard for Internet Calendaring and Scheduling," press release, Netscape Communications, 24 July 1996; http://wp.netscape.com/newsref/pr/newsrelease194.html.
2. F. Dawson and D. Stenerson, *Internet Calendaring and Scheduling Core Object Specification (iCalendar)*, RFC 2445, Nov. 1998; www.ietf.org/rfc/rfc2445.txt.
3. S. Silverberg et al., *iCalendar Transport-Independent Interoperability Protocol (iTIP) — Scheduling Events, Busy Time, To-Dos, and Journal Entries*, RFC 2446, Nov. 1998; www.ietf.org/rfc/rfc2446.txt.

4. F. Dawson, S. Mansour, and S. Silverberg, *iCalendar Message-Based Interoperability Protocol (iMIP)*, RFC 2447, Nov. 1998; www.ietf.org/rfc/rfc2447.txt.
5. M. Rose, *The Blocks Extensible Exchange Protocol Core*, RFC 3080, Mar. 2001; www.ietf.org/rfc/rfc3080.txt.
6. Y. Goland et al., *HTTP Extensions for Distributed Authoring — WebDAV*, RFC 2518, Feb. 1999; www.ietf.org/rfc/rfc2518.txt.
7. C. Daboo, B. Desruisseaux, and L. Dusseault, "Calendaring and Scheduling Extensions to WebDAV (CalDAV)," IETF Internet draft, Feb. 2005; work in progress.
8. G. Clemm et al., *Web Distributed Authoring and Versioning (WebDAV) Access Control Protocol*, RFC 3744, May 2004; www.ietf.org/rfc/rfc3744.txt.
9. G. Clemm et al., *Versioning Extensions to WebDAV*, RFC 3253, Mar. 2002; www.ietf.org/rfc/rfc3253.txt.

Lisa Dusseault is a development manager and standards architect at the Open Source Application Foundation, where she is responsible for email, calendaring, and general sharing protocols for Chandler, a personal information manager. Dusseault is currently cochair of the IETF WebDAV working group as well as the iMap Extensions WG, and she participates regularly in other IETF work particularly relating to HTTP and Instant Messaging (XMPP). She has a BS in systems design engineering from the University of Waterloo. Contact her at lisa@osafoundation.org.

Jim Whitehead is an assistant professor in the Department of Computer Science at the University of California, Santa Cruz. He founded the IETF WebDAV WG and served as its chair from its inception in March 1997 through March 2004. His research interests include collaborative authoring, software configuration management, software evolution, and Web engineering. Whitehead received a PhD in information and computer science from the University of California, Irvine. He is a member of the ACM, Usenix, and the IEEE. Contact him at ejw@cs.ucsc.edu.