Automation for Calendaring

Date: 2013-04-24
Version: $Id:$
Author: Rick van Rein
Contact: rick@openfortress.nl
Organization: OpenFortress BV

Dedication

On assignment from NLnet.nl

Abstract

One of the many unfulfilled promises on the Internet is to have a distributed solution for semi-automatic calendaring. In spite of the existence of calendaring standards, it still takes a lot of effort to schedule a time together in a distributed fashion. This proposal uses agents for scheduling and responding to get the best out of the existing standards of today.

Contents

Introduction

Calendaring is not a new issue, and quite a few steps have been taken to solve the problem of planning some time together between people who are scattered accross the Internet, where they are distributed in space and time.

The iCalendar standard defines a format for exchanging calendaring information, and iTIP defines a protocol for exchanging it without defining the precise transport mechanism. Concrete mechanisms for the transport of iCalendar data are iMIP (using email) and WebCal (using HTTP). The WebCal services can be located under a domain using SRV/TXT records.

This setup gives possibilities to attempt an HTTP upload to a WebCal service or, failing that, to send an email with an iCalendar body or attachment. Software can pickup on these files automatically thanks to the `text/calendar` MIME-type, and

process it locally, which leads to fairly good desktop integration if users have setup calendaring, or are at least cabable of processing such files and responding to them after checking their paper diary. Moreover, using the `multipart/alternative` MIME-type for the body, it is also possible to send a human-readable version of the calendar if a local iCalendar helper application is not setup.

All this would seem to make planning and scheduling a sneeze, but not in reality. Not all recipients respond to calendaring requests if they do not keep a digital agenda. The HTTP method seems nice but has rather crude authentication facilities -- certainly not OpenID authentication and OAuth authorization.

Furthermore, the entire processing part seems to rely on human action. Inasfar as calendaring software links, it usually comes down to polling over WebCal, leading to reduced interactivity compared to the email method, but an email takes more effort.

We have seen web services jump into this gap, by offering a number of alternate dates and times for an upcoming appointment, and let potential participants fill out when they could make it. This is a step forward. The disadvantage is that it is not located under one's control, so the advantages from automation of the process require adding contact details for those who should attend, and that would imply loss of privacy. The best solution therefore is to run a similar service under one's own domain, and use the contact information (such as whitelists under a pseudonym) to quickly schedule meetings and to interact automatically with the participants.

Towards a Solution

The SRV/TXT records for WebCal location discovery is a big step forwards, but common software merely stores calendaring requests and does not do anything clever with them. Ideally, a human would be involved, with as much guidance as possible by software.

Within NEA, we tend to separate outgoing and incoming services; this has a straightforward mapping to calendaring, where the outgoing service would be planning and scheduling support, and the incoming service would be finding out if a proposal can be met. Each of these processes could be guided by an agent.

One type of agent that comes to mind for this is XMPP; it has the capability of sending simple forms, and may use those to get quick responses from users who are actively online. And if someone is not online, a fallback to a later XMPP inquiry or an email message still exists.

Outgoing iCalendar: Scheduling Agent

An outgoing agent would permit setting up a meeting plan, much like is done with the aforementioned web services, and present the options to the various users over discovered channels; using WebCal if suggested with SRV/TXT records and if permitted, and otherwise falling back to email.

Any impulse to send an XMPP form directly to a connected XMPP peer must be suppressed, as that would deprive them of their ability to process iCalendar data with their own (possibly NEA) calendaring solution. Any cross-domain communication, and in fact any cross-user communication, should be done through standard iTIP mechanisms, so iMIP and WebCal. Note that it might make sense if an iTIP mechanism over XMPP was considered.

Note that web services that present scheduling alternatives have inspired iCalendar support to poll for suitable times through a VPOLL extension. It encapsulates a number of alternative VEVENT proposals. See draft-york-vpoll for more information.

The outgoing service should report back to the organiser of a meeting at a number of opportunities, which may be setup as preferences:

- When an invitee responds to the invitation;
- When someone other than an invitee responds to the invitation;
- When a prescheduled time has passed;
- When all invited participants have responded;

It is also likely that outgoing services would consult WebDav repositories to check availability before presenting the user with a display in which to make proposals.

Incoming iCalendar: Constraint Guardance and User Interaction

An incoming service for calendaring should pickup iCalendar information from a number of iTIP sources; currently, these include iMIP email and WebCal over HTTP. It would also help to respond to any such inquiries, especially when it has to be presented to the user in a simple HTML format.

Note that it is not unlikely that an XMPP extension will also be defined to implement iTIP. Such an extension might cause calendaring requests to be routed to the desktop, and raise the proposed event in a local calendaring application, and have it handle the responses.

Inasfar as an incoming service sees no obstructions to forward a proposal interactively to a user, it can redirect a request over interactive channels such as XMPP, and ask the user for feedback on a proposed meeting. An attachment with the iCalendar file could be sent alongside, to support instant integration with desktop calendaring tools.

A more useful approach might be to use a publish/subscribe mechanism, such as an ATOM feed or an XMPP service, and relay iCalender information over that channel. A calendar application on a desktop usually employs polling, but publish/subscribe mechanisms are less resource-intensive and they can be instantaneous in presenting the request in a local calendar by opening the iCalendar file in it through whatever desktop mechanism is available. An XMPP service is particularly interesting to this end, because it can be discovered on the fly and would relay to the right client. Note that an equivalent XML representation for iCalendar also exists. TODO: Can XMPP Service Discover / Feature Negotiation support iCalendar endpoint discovery?

One facility that is useful for an incoming iCalendar service is of course guarding any clash between appointments. It is possible that users want to setup sets that may, and others that may not overlap. A suitable model should be found, and it will probably become a hierarchy even though it is in reality probably a poset. This is useful to express that any plans in diary A may overrule those in diary B, but not conversely. This may however be handled by setting certain meetings as TENTATIVE at best, for example private chores during working hours or unpaid overtime work in the evening.

It is possible that users will want to make such dependencies time-related: in the evenings my private plans prevail, while at daytime the job is overruling. The best way of handling this is perhaps to define a maximum or default confidence level for plans in one agenda during a certain time. This facilitates a rough allocation of time to certain activities, and leaves open the option of shifting plans in non-standard ways.

Any planned meetings in the agenda can be used to respond immediately to any free/busy inquiries, as well as to attempts to schedule a meeting at an already-taken time.

Note that relations to other agendas must include relations to remote agendas. One might want to take a shared agenda with a partner into account, or a company agenda, or one might simply be involved with multiple companies. In the same way, an agenda should be opened up to other parties; the NEA design can make good use of whitelists and pseudonyms for that purpose.

Note that an export of an agenda may also incorporate busy times caused by external agenda's, and that could lead to cyclic dependencies. The incorporation of idempotence into such relations is helpful to that end. Also, an ability to subscribe to an agenda is helpful; any publish/subscribe mechanism that can send `text/calendar` MIME-types is in fact suitable to that end, as long as peers can agree to it.

A final practicality of an incoming service is that it should take setup/unwind times into account. For instance, a meeting may require a bit of preparation time, and perhaps some time to round it off properly. Also, travel time may be involved. All these factors would make a meeting in an agenda appear as a longer meeting than the one that is being proposed.

Connection between Incoming and Outgoing iCalendar Services

There is no strict requirement to send responses to iCalendar invitations over the same iTIP channel as the original request. For instance, a request that enters over WebCal might be routed over XMPP to a calendaring tool on the desktop, which sends its response over iMIP email. In the end, the iCalendar files should meet in an agent that handles it all.

Given the split of incoming and outgoing iTIP channels, it is important to realise that responses may well return over another channel than the outgoing channel, for example when a slow user responds after a temporary connection was torn down. To make this possible, the incoming iCalendar service should pick up on messages with a local organiser, and relay any such messages to the outgoing iCalendar service.

This means that the incoming and outgoing service need to be connected in a manner that surpasses the usual discovery methods. One way for doing this is to hardcode such connections; another is to use an alternatively named SRV/TXT record set to find the response address for an outgoing WebCal service. TODO: Or is WebCal response already taken care of in the WebCal RFC?

Subscription to Calendaring through Publish/Subscribe

Calendaring information have the potential to drive other tools, and thereby improve automation. Think of call forwarding setups, that relay to an attendant while a meeting takes place; think of automatic activation of a phone meeting facility with a calendar-driven access code; think of XMPP presence information based on calendar availability; think of this driving the decision when to hold a (long) SIP phone call with someone.

Each of these applications trives on subscriptions to calendaring information, and driving their choices by it. In terms of NEA, this could simply be reduced to attributes that indicate whether the user is currently scheduled to be in a meeting of some kind.

These attributes then drive the import/export processes. The free/busy attributes may have to be stretched to cover whether a particular pseudonym is just busy in

some opaque manner, or whether a user is actively involved with someone within that pseudonym. Other users from the meeting, and perhaps anyone in the pseudonym would be quite welcome to communicate while in a meeting with them!