USENIX Association

Proceedings of the 4th Annual Linux Showcase & Conference, Atlanta

Atlanta, Georgia, USA
October 10–14, 2000

Open Source Group Calendaring: GCTP and OpenFlock

David L. Sifry
Chief Technology Officer
Linuxcare, Inc.
650 Townsend Street, San Francisco, CA 94103.
dsifry@linuxcare.com, http://www.linuxcare.com/

Abstract

This paper describes a new protocol and open–source system that is designed to solve a significant problem — how to manage and organize individual and group schedules in a cross-platform manner. In addition, the design was created to allow for interoperability and extensibility in the protocol and functionality of the system, and discussion is given on how to create an extensible protocol while maintaining backwards compatability.

1 Introduction

While many proprietary solutions exist for group calendaring (Microsoft Outlook/Exchange, Lotus Notes, Novell Groupwise, etc) there were no free software-based groupware tools available. In addition, while the IETF working group on calendaring is working on interoperability standards for sharing contacts and information between different LAN-based calendaring systems, there was no work being done on a free alternative to the LAN-based calendaring systems. Two standards, vCard (iCard) for the exchange of contact information, and vCalendar (iCalendar) for the exchange of schedule information have been proposed, but these proposed standards only offer a solution for transport between different systems - they do not take into account a variety of functions that are as important, such as the method of transport, methods of securing or rejecting appointments, and methods of rescheduling appointments. vCalendar and vCard are nouns in the group scheduling vocabulary - but they do not describe the verbs. What became necessary was an open way to describe both nouns and verbs, and an implementation or set of implementation that met the needs of most users of the group scheduling system.

2 What is OpenFlock

OpenFlock is a group calendaring system. It handles calendars, projects, and teams in much the same way that Sendmail handles email, or BIND handles DNS. OpenFlock is a GCTP (*Group Calendaring Transport Protocol*)-compliant system, and is actually the first such system in existence, as the GCTP protocol was designed in parallel with OpenFlock's development.

3 OpenFlock Architecture

The OpenFlock architecture is client-server using GCTP as the protocol between the calendaring server and client. Each connection is authenticated, much in the same way that POP3 or IMAP connections are authenticated, however, a plugin authentication layer is used to allow for extensibility. OpenFlock is designed to be able to export and import vCalendar objects to increase interoperability with other vCalendar-capable schedulers. By using GCTP however, much richer functionality is possible. For example, it is possible to ask the server to schedule an appointment with a predefined team of people for one hour during business hours in the upcoming week, and the GCTP server will find the first available time period that meets those conditions. In addition, the system allows for the concept of a proxy, that is, someone who is capable of doing all the things the user that she is proxy for is capable of doing. This is applicable for administrative assistants at a company, for example. The system is also set up with arbitrary access control capabilities, so that, for example, bosses can auto-schedule appointments with subordinates, and can view subordinates' non-private schedule entries. A queue exists for each user, and whenever a change is made to a person's schedule without her knowledge, a notification is added to her queue, along with an opportunity to confirm or reject the change. In this way, the GCTP server maintains a set of knowledge of what appointments have been confirmed or rejected, and if necessary, it can trigger a rescheduling of an appointment if one or more of the attendees rejects the original time or date.

The application is also designed in such a way that a user can synchronize her client for off-line use, can queue up scheduling changes, and upon reconnection to the server, request the changes in batch. This synchronization capability is built into the protocol itself. It also allows users to synchronize group calendars before going on a trip on a personal laptop, and then synchronize individual calendars on a handheld device, such as a palm pilot. In addition, the design accomodates a palm-based client application with full functionality that can log on, authenticate, and then synchronize calendars and contact lists with the OpenFlock server directly.

3.1 Plugin architecture

OpenFlock has plugin capability for its authentication modules. Currently, passwd, shadow password, MD5-challenge-response, and LDAP authentication is supported. More is forthcoming.

OpenFlock also implements a form of Access Control Lists (ACLs) on top of its database schema. This ACL system is pluggable, allowing different storage backends. Currently, SQL database (PostgresSQL and MySQL in particular) persistent storage is supported, but others can be implemented.

4 The GCTP Protocol

The GCTP protocol is described in full at http://www.gctp.org/protocol.txt but a selection of commands is described below. I have intentionally left out most of the real nitty-gritty of the protocol in this paper, like the regular expression syntax, all of the possible error conditions for each command, etc. These are enumerated in the protocol description listed above. It should also be notes that the protocol is a work in progress — What is described here is protocol version 1.0.

4.1 Environmental Commands

USER *username*

Specifies the username for the authentication session

LIST AUTHMETHODS

Returns a list of authentication methods that the GCTP server accpts. All GCTP servers must accept the NONE, PASSWD, and RANDOM-MD5 authentication methods at a minimum. For example, here is the conversation for a minimal server:

```
>>>LIST AUTHMETHODS
<<<2
<<<OK PASSWORD
<<<OK RANDOM-MD5
```

LIST PEOPLE

Provides a list of people on the system.

LIST LOCATIONS

Provides a list of locations on the system.

LIST CAMPUSES

Provides a list of campuses on the system.

LIST PROJECTS

Provides a list of projects on the system.

LIST TEAMS

Provides a list of teams on the system.

AUTHMETHOD *authentication-method*

Specifies the authentication method used for this session.

AUTH *string*

Sends authentication information to the server. There may be a number of send-reply cycles within the authorization sequence. For example, if the password is "password" and the AUTHMETHOD is RANDOM-MD5, then the sequence will look like this:

```
>>>USER user
>>>AUTHMETHOD RANDOM-MD5
<<<OK daopkaepkoeakope3k
>>>AUTH 1qdpc/e5dVmuvDLH+rGc0A
<<<OK AUTHORIZED
```

4.2 Connection Control Commands

RSET

Reset the connection to a new, unauthorized state. Useful for keeping connections alive without having to break down the socket.

MODE *outputmode*

Changes the way that the server delivers output. There are currently two settings:

MODE TEXT

Causes the server to send output as plain text, not as the default Base64. This is for testing purposes - using TEXTMODE can be difficult to parse, as embedded newlines, etc are not delimited.

MODE BASE64 Causes the server to send output as Base64-encoded text, which is the default.

4.3 Appointment Commands

NEW APPT *appointment-string*

Sends a request to the server to create a new appointment. The *appointment-string* is a block of XML markup, describing the appointment. Valid types of appointments currently are vCAL, iCAL, and INTERNAL (the OpenFlock internal format, used for debugging purposes). The INTERNAL format is optional in GCTP implementations - but vCAL and iCAL are mandatory. Group appointment requests are an integral part of this request - including requests for appointments where certain items, like the date or time of the appointment, are left blank. This causes the GCTP server to find the earliest time when all people who are marked as *REQUIRED* for the appointment are available to attend.

Possible responses include:

```
OK Confirmed id
OK Tentative id
```

These denote confirmed and tentatively confirmed appointments - the Unique ID for each appointment is returned also.

Possible Error conditions include:

```
ERROR Authentication
ERROR Permissions
ERROR User
ERROR Impossible
ERROR BadData
```

Each one of the possible return conditions are described in more detail in the GCTP protocol description, they are listed here as a description of the variety of return conditions that must be dealt with in a group calendaring protocol.

GET APPT *apptid*

Sends a request to the server to get information on the requested appointments. If there are no errors, the server will respond with a block of XML markup describing the appointment with the *apptid* identifier.

MODIFY APPT *apptid appointment-string*

Sends a request to the server to modify an appointment with the unique identifier *apptid*.

DELETE APPT *apptid*

Deletes the appointment with the unique appointment ID *apptid*.

4.4 Search Commands

SEARCH PEOPLE *arg1 expr regexp*

Returns a list of people on the system where *arg1* is:

```
FIRSTNAME
MIDDLENAME
LASTNAME
NAME
TITLE
```

```
PHONE
FAX
EMAIL
ADDRESS
```

expr is: '==' or '!=', denoting equals and not equals, and *regexp* is a regular expression.

Possible responses include:

SEARCH LOCATIONS *arg1 expr regexp*

Returns a list of locations on the system where *arg1* is:

```
CAMPUS
BUILDING
ROOM
ADDRESS
```

expr is: '==' or '!=', denoting equals and not equals, and *regexp* is a regular expression.

Possible responses include:

```
OK string
ERROR Authentication
ERROR BadData
```

SEARCH PROJECTS *arg1 expr regexp*

Returns a list of projects on the system where *arg1* is:

```
NAME
DESCRIPTION
```

expr is: '==' or '!=', denoting equals and not equals, and *regexp* is a regular expression.

Possible responses include:

```
OK string
ERROR Authentication
ERROR BadData
```

SEARCH TEAMS *arg1 expr regexp*

Returns a list of teams on the system where *arg1* is:

```
NAME
DESCRIPTION
```

expr is: '==' or '!=', denoting equals and not equals, and *regexp* is a regular expression.

Possible responses include:

```
OK string
ERROR Authentication
ERROR BadData
```

4.5 Schedule Commands

GET SCHEDULE *uid date1 date2*

Retrieves a list of appointments for the user with UID *uid*, starting at *date1* and ending at *date2*. Dates are represented as follows:

```
1999-02-15 05:00:00 = February 15, 1999 05:00:00 AM GMT.
2000-10-12 23:12:15 = October 12, 2000 11:12:15 PM GMT.
```

All dates are stored in GMT.

GET NOTIFICATIONS *uid*

Retrieves a list of notifications and appointment requests for the user *uid*.

CONFIRM *uid apptid*

Confirms the appointment with ID *apptid* listed for user *uid*.

REJECT *uid apptid*

Rejects the appointment with ID *apptid* listed for user *uid*.

GET PERSONLIST *listid*

This command retrieves the person id's that make up the personlist *listid* (used in the group scheduling process).

4.6 Miscellaneous Commands

HELP

This command will list the top-level commands available on the server.

5 GCTP design guidelines

The three design guidelines of the GCTP protocol are:

Evolvability

Plugin architecture

Gracefully handle unknowns

5.1 Evolvability

This guideline is probably the most important part of the design criteria, because I anticipate that as GCTP sees additional use, new commands will be added, new command formats will be added, and the data description will change. Given these inevitabilities, it was important to design for evolvability right from te start. One of the main syles of design I used was a tuple–based mechanism to describe commands. As GCTP stands now, it only has 13 base commands, listed below:

```
USER
AUTHMETHOD
AUTH
RSET
LIST
MODE
NEW
GET
MODIFY
DELETE
SEARCH
CONFIRM
REJECT
```

Of those commands, 7 (LIST, MODE, NEW, GET, MODIFY, DELETE, SEARCH) already have subcommands, i.e. "SEARCH PEOPLE". This use of sub-commands illustrates the design for modularity and extensability. Adding a new SEARCH command, for example, need not affect other parts of the protocol, and from an implementation (coding) perspective, it allows the new subcommand code to be self-contained and modular.

5.2 Plugin architecture

OpenFlock also supports a generic plugin architecture. This set of APIs is under active development, and a set of authentication plugins that implement a variety of schemes are available. The back-end storage structure is also pluggable, so although OpenFlock currently uses a SQL database for its persistent storage, other avenues are available, such as a filesystem-based approach.

5.3 Gracefully handle unknowns

The third main architecture guideline is to gracefully handle unknowns. Since GCTP is expected to go through a number of protocol revisions in a relatively quick manner, I designed for backwards compatability from the start. If a new GCTP client connects to an older protocol revision server, it will fail gracefully on the new protocol features but still be able to interact with the server's set of commands. This allows for a more smooth transition of system upgrades. The hope here is that a new protocol revision will not instantly obsolete older protocol revisions, but will be an extension to the older revisions.

6 Current Status of the project

The GCTP protocol is approaching 1.0 stability, and will probably be at 1.0 by the time this paper is published. A few additional features are being worked on at this writing, namely, the NEW APPT syntax that allows the server to schedule an appointment with a predefined team of people for one hour during business hours in the upcoming week. The most up-to-date information on the protocol, and servers and clients implementing the protocol is at `http://www.gctp.org/`. Currently, work is ongoing to add GCTP support to Helix Code's *Evolution* project, and work on a pilot-based client is underway in addition to the OpenFlock server. A PHP-based web client is also being built.

7 Future directions

After the 1.0 release of the GCTP protocol and OpenFlock, there is much to be done. One of the first priorities is to fully integrate GCTP support into open source applications, like office suites and PIMs. Work on PDA synchronization for offline use is also a priority. In addition, we're looking for developers with MAPI experience, so that a GCTP MAPI conduit could be made available for people who are using MS Outlook and Exchange. Multiple transport layer support (GCTP over SMTP, for example) will help people who are not always connecting from a LAN environment or who don't want to allow socket-based connections to their GCTP server for whatever reason. More work is ongoing, and a mailing list exists for developers, protocol-dev@gctp.org is the mailing list for protocol development of the GCTP protocol, and devel@openflock.org is the mailing list for OpenFlock developers.

8 Acknowledgments

Many people have approached me with ideas and have helped in the design of GCTP and OpenFlock. In particular, I'd like to thank David Desrosiers, Rasmus Lerdorf, Luke Leighton, David Mandala, and Andrew Tridgell. I'd also like to thank all of the folks at Linuxcare who have worked tirelessly to provide a place where open source development is encouraged and funded.

9 Availability

OpenFlock is free software, licensed under the GPL. It is available via anonymous FTP from

`ftp://ftp.openflock.org/pub/openflock/`

Information, including anonymous CVS access, is also available on the OpenFlock homepage at

`http://www.openflock.org/`

The development mailing list for OpenFlock is `devel@openflock.org`.

Information on the GCTP protocol is available at

`http://www.gctp.org/`

The development mailing list for GCTP protocol development is `protocol-dev@gctp.org`.