Internet Calendaring and Scheduling Core Object Specification (iCalendar) Compatible Collaborative Calendar-Server (CCS) Web Services

Indiana University Computer Science Department,
Bloomington, IN, 47405

Ahmet Fatih Mustacoglu and Geoffrey Fox

{amustaco, gfc}@cs.indiana.edu

Abstract

It is very obvious that Internet needs to have interoperable calendaring and scheduling services to provide interoperability between different types of calendaring and scheduling applications. Internet Calendaring and Scheduling Core Object Specification (iCalendar) has been defined to provide the definition of a common format for openly exchanging calendaring and scheduling information across the Internet. In the first part of this report, we overview the internet calendaring and survey the software development efforts and applications that are using iCalendar specifications. We therefore, in the second part of this report, describe our efforts to implement Collaborative Calendar-Server Service (CCS) as web services, and find the applicability of Web Service Architecture on to Collaborative Calendaring using iCalendar standards.

We have also built a bridge module that allows our Collaborative Calendar-Server (CCS) to communicate with different calendar and scheduling applications that supports iCalendar standards. This module receives the requests from clients, and executes the associated services of Collaborative Calendar-Server (CCS) based on the coming requests.

1 Introduction

Internet Calendaring and Scheduling Core Object Specification (iCalendar) introduces a new format for calendaring and scheduling applications. Without a common format, there will be interoperability problems between dissimilar applications that are not supporting the same format for defining the calendar information. Different organizations and commercial vendors develop their own calendaring and scheduling model and structures. If the calendar representation format is not interoperable, calendar applications can not communicate with each other even if though they are in the same organization or they are coming from the same commercial vendor.

To solve interoperability problems, The Internet Engineering Task Force (IETF) [17] introduced some standards by publishing specifications for the calendar data exchange. IETF is an open community of network designers, operators, vendors, and researchers for every interested individual [17].

This document gives the details about the design, the architecture, and the implementation details of our Web Service oriented Collaborative Calendar-Server that supports iCalendar standards. Our calendar server has also a bridge module that enables other calendaring and scheduling clients, which use http methods to call our services, to communicate with our calendar server such as Mozilla Calendar Client. In this document, we will first mention about the some calendaring technology terms, definitions and related works for implementing calendaring and scheduling applications that supports iCalendar standards. Next, we are going to mention about Internet Calendaring and Scheduling Core Object Specification (iCalendar). And then, we discuss our Collaborative Calendar-Server implementations details, which include the detailed explanation of the architecture, the design, GlobalMMCS Client Module for CCS, implementation of web services of Collaborative Calendar-Server, and the bridge module for CCS, and we will talk about the applicability of web service architecture on to CCS using iCalendar standards. Finally, we will provide the conclusion part as a last chapter.

2 Guide to Internet Calendaring and Related Work

2.1 Calendaring Technology and Terms and Definitions

Calendaring and scheduling protocols provide organizations or individuals with a way to obtain calendaring information and to schedule meetings across the Internet to progress the level of interoperability possible between dissimilar calendaring and scheduling applications. RFC 2445 (iCalendar Specification Protocol) for instance provides the definition of a common format for openly exchanging calendaring and scheduling information across the Internet. iCalendar is the language to describe the calendar objects.

iCalendar (RFC-2445) defines a data format for representing calendar information, in order to be used and exchanged by other protocols. iCalendar (RFC-2445) can also be used in other contexts, such as an export/import feature or a drag-and-drop interface. All the other calendaring protocols depend on iCalendar (RFC-2445), so all elements of a standards-based calendaring and scheduling systems will have to be able to translate iCalendar (RFC-2445) [3].

The following definitions are the common terms for the Internet Calendaring.

Calendar: Calendars maintains the calendaring information as a storage container. A calendar consists of events, to-dos, tasks, journal entries, etc. A calendar basically represents a person's specialized need, or his/her agenda.

Calendar Access Rights: It is a set of rules that specifies the user's permissions on what kind of operations a user can execute on the calendar. Such as read or write rights on a given calendar.

Calendar Service: It is a server application and it provides access to a number of calendar stores.

Calendar Store (CS): A calendar service's date is stored in a storage called Calendar Store. A calendar store may store one or more calendars, properties and components other than the calendars.

Calendar User (CU): It is an entity that access and use the calendar data. It is usually a human.

Calendar User Agent (CUA): A calendar user communicates with the calendar service or calendar storage through software and this software is called as Calendar User Agent.

Component: Component is a part of calendar data such as a to-do, an event, or a task, etc. Properties are used to keep the information about components.

Delegator: A calendar user who has assigned his or her participation in a scheduled calendar component (e.g. a VEVENT) to another calendar user (sometimes called the delegate or delegatee). An example of a delegator is a busy executive sending an employee to a meeting in his or her place [3].

Delegate: A calendar user (sometimes called the delegatee) who has been assigned to participate in a scheduled calendar component (e.g. a VEVENT) in place of one of the attendees in that component (sometimes called the delegator). An example of a delegate is a team member sent to a particular meeting [3].

Designate: Designate is a calendar user, and he or she has been authorized to act on behalf of another calendar user.

Local Store: It is a calendar store that is on the same device with a calendar user agent.

Property: Property describes some elements of a component such as start time, end time, summary, time zone etc.

Remote Store: It is a calendar store that is not on the same device with a calendar user agent.

2.2 Related Works

2.2.1 Collaborative/Non-Collaborative Calendaring and Scheduling Implementation Efforts

This chapter lists some collaborative or non-collaborative calendaring and scheduling applications and servers that are compatible with Internet Calendaring and Scheduling Core Object Specification (iCalendar) [1]. Some of the specifications are abstract some of them are just discussion papers. We have not mentioned from them in this chapter.
Here you will see some application efforts that can support Internet Calendaring and Scheduling Core Object Specification (iCalendar) standards.

Hula Project: Hula is an open source project and it has been led by Novel. It is a calendar and mail server, and available at: http://hula-project.org/Hula_Project.

PHP iCalendar: PHP iCalendar can display the iCal files in a web browser. It is a PHP based iCal file viewer and parser. It is based on IETF Specification (iCalendar). Available at: http://phpicalendar.net/documentation/index.php?title=Main_Page.

phpGroupWare: It is written in PHP, and it is a multi user groupware suite. It provides many web-based applications such as Calendar, To-do List, Notes, Addressbook, and Newsgroup etc.
Available at: http://www.phpgroupware.org/ .

Sun Java System Calendar Server: The Sun Java System Calendar Server makes the team collaboration easier by enabling users to manage and coordinate their appointments, tasks, and resources. Available at: http://www.sun.com/software/products/calendar_srvr/index.xml .

Cyber Scheduler: Cyber Scheduler is a web-based calendaring and scheduling solution. Available at: http://www.crosswind.com/cybdata.htm .

Java iCal Group Scheduler: It is an open source project developed by SourceForge.net in Java language. Java iCal Group Scheduler let users schedule meeting automatically. Available at: http://sourceforge.net/projects/jical/ .

ScheduleWorld: ScheduleWorld is both a secure Internet calendaring and scheduling client application and a server service that enables users to schedule events or appointments with other ScheduleWorld users and/or users of other calendaring and scheduling products [2].
Available at: http://www.scheduleworld.com/ .

Kronolith Calendar Application: Kronolith is the Horde calendar application. It provides a stable and featureful individual calendar system for every Horde user, with integrated collaboration/scheduling features. It makes extensive use of the Horde Framework to provide integration with other applications [4]. The Horde Application Framework is a general-purpose web application framework in PHP, providing classes for dealing with preferences, compression, browser detection, connection tracking, MIME handling, and more [5].
Available at: http://www.horde.org/kronolith/.

phpMyCal: It is a shared calendar that iCal can subscribe to and all users can add items to from a Web interface. This is not a full calendaring solution yet, since users won't be able to make changes from iCal itself, they can just view the calendar. The actual calendar is kept in a MySQL database that dynamically creates the iCal data files when queried by clients [6].
Available at: http://dev.neb.net/phpMyCal/ .

Favorin Time: Favorin Time is a client and a server group calendar for Linux and other UNIX environments with KDE desktop. They have a free open source client available for download.
Favorin Time Server is a commercial product that is sold separately, and it provides groupware functionality for Favorin Time client [8].
Basically, it provides calendar management and scheduling for organizations. Available at: http://www.favorin.com/.

Web Organizer: Web Organizer (WEBO) is a groupware solution with a calendaring module. Available at: http://weborganizer.sourceforge.net/ .

Web Calendar: Web Calendar is a PHP-based calendar application, and it is used for maintaining calendars for single or multiple users. It supports iCalendar and can communicate the iCal-compliant calendar program such as Apple's iCal, Mozilla Calendar or Sunbird. Available at: http://www.k5n.us/webcalendar.php .
Chronos: Chronos is a Web agenda/calendar for Intranets (even if it can be used from anywhere). It can send reminders by email. You can schedule multi-user events. It is fast and light on resources (the balance size/speed can be tweaked by tweaking mod_perl and Apache) [9].
Available at: http://chronoss.sourceforge.net/ .

MRBS (Meeting Room Booking System): MRBS is a free, GPL, web application using PHP and MySQL/pgsql for booking meeting rooms. It's similar in concept to Netscape Calendar [10]. Available at: http://mrbs.sourceforge.net/ .

Meeting Maker: It is a collaborative calendaring and scheduling application for organizations. It provide users with seeing other users' availability, sending invitations to others, accepting or declining meetings etc.
Available at: http://www.meetingmaker.com/products/meetingmaker/default.cfm .

OpenCAP: It is an open source calendar server based on iCalendar specifications. Its development is still in progress.
And available at: http://www.kiv.zcu.cz/~simekm/calendar/OpenCAP/doc/ .

UW Calendar: The UW Calendar project is building an open-source calendaring system for higher education. UW Calendar will support personal, public and group events, use existing open standards, and support web-based and other forms of access, including uPortal integration [11].
Available at: http://www.washington.edu/ucal/ .

3 Internet Calendaring and Scheduling Core Object Specification (iCalendar)

Internet Calendaring and Scheduling Core Object Specification (iCalendar) enables developers to create interoperable components. iCalendar specifications defined in the RFC2445 are available at: http://www.ietf.org/rfc/rfc2445.txt .

3.1 iCalendar Specification Basics (RFC 2445)

The calendaring and scheduling need has been increasing rapidly for last decade. Companies have been trying to implement this technology into their businesses. Unfortunately, there is a lack of Internet standards for the message content types that are crucial to calendaring and scheduling applications. iCalendar standards defined in RFC2445 has intended to advance the interoperability among the calendars and scheduling applications that are not similar.

The iCalendar specification is a result of the work of the Internet Engineering Task Force Calendaring and Scheduling Working Group (chaired by Anik Ganguly of Open Text Inc.), and was authored by Frank Dawson of Lotus Development Corporation and Derik Stenerson of Microsoft Corporation. iCalendar is heavily based on the earlier vCalendar industry specification by the Internet Mail Consortium (IMC) [15].

iCalendar specifications define the format for specifying iCalendar object methods. An iCalendar object method is a set of usage constraints for the icalendar object [1].

The iCalendar format is suitable as an exchange format between applications or systems. The format is defined in terms of a MIME content type text/calendar. The “ics” file extension is used to indicate a file containing (an arbitrary set of) calendaring and scheduling information consistent with this MIME content type [15]. This will enable the object to be exchanged using several transports, including but not limited to SMTP, HTTP, a file system, desktop interactive protocols such as the use of a memory-based clipboard or drag/drop interactions, point-to-point asynchronous communications, wired-network transport, or some form of unwired transport such as infrared might also be used [1].

3.2 iCalendar Core Object

Calendaring and Scheduling Core Object is the top-level object in iCalendar, and it is a collection of calendaring and scheduling information. Usually, this information consists of a single iCalendar object, but two or more iCalendar objects can be grouped together consecutively. The first line must be "BEGIN: VCALENDAR", and the last line must be "END: VCALENDAR"; the contents between these lines is the body of the iCalendar object and called the "icalbody". The icalbody consists of a sequence of calendar properties and one or more calendar components. The calendar properties are attributes that apply to the calendar as a whole. The calendar components are collections of properties that express a particular calendar semantic. For example, the calendar component can specify an event, a to-do, a journal entry, time zone information, or free/busy time information, or an alarm [15].

Here is a simple example (from RFC 2445) of an iCalendar object that defines a "Bastille Day Party" event occurring from July 14, 1997 17:00 (UTC) through July 15, 1997 03:59:59 (UTC):

```
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//hacksw/handcal//NONSGML v1.0//EN
BEGIN:VEVENT
DTSTART:19970714T170000Z
DTEND:19970715T035959Z
SUMMARY:Bastille Day Party
END:VEVENT
END:VCALENDAR [15].
```

The iCalendar components defined in RFC 2445 can be listed as below:

- Events (VEVENT): This component provides a grouping of component properties that describe an event that represents a scheduled amount of time on a calendar [15].
- To-do (VTODO): This event represents a to-do item on a calendar, i.e., an action-item or assignment [15].
- Journal Entry (VJOURNAL): This component describes a journal entry on a calendar.
- Free/Busy Time (VFREEBUSY): A VFREEBUSY component is used for describing either a request for free/busy time, which describes a response to a request, or describing a published set of busy time [15].
- VTIMEZONE: This component is used for defining Time Zone information. And, it is usually used for supporting other components [15].
- VALARM: This component is used for defining alarms, and it is included in other components most of the time [15].

4 Collaborative Calendar-Server (CCS) Implementation and Applicability of Web Services

In Chapter 3 we give general information about the Internet Calendaring and Scheduling Core Object Specification (iCalendar) specifications. Here in this chapter there will be more detailed information about the implementation of our Collaborative Calendar-Server project.

4.1 Collaborative Calendar-Server (CCS) Design and Architecture

Collaborative Calendar-Server (CCS) enables users to manage and coordinate their appointments, events, and tasks. With its user friendly web-interface integrated into GlobalMMCS portal, end users can access their personal calendar (private calendar) or group calendar (collaborative calendar) from anywhere, anytime by using web browser. Users can also update, or schedule a new event into their private calendar or into the group calendar. Furthermore, there is a bridge module that enables user to publish or to subscribe to their calendar or to collaborative calendar using Mozilla Calendar client application. For Architecture of Collaborative Calendar-Server, please see Figure 1.

Collaborative Calendar-Server implemented as a web service in Java programming language, and it uses iCal4j Java library [16] for reading and writing iCalendar data streams as defined in RFC2445. It stores the users' calendar and collaborative calendar in iCalendar format (with "ics" extension) as specified in RFC 2445 on the machine where the Collaborative Calendar-Server is running. Each user has their own iCalendar file for their private calendar. And, there is only one iCalendar file for collaborative calendar shared by all users. Users' access to collaborative calendar has been synchronized.

The bridge module has been implemented as a java servlet, and it enables various calendar and scheduling clients, which support iCalendar specifications, to communicate with our Collaborative Calendar-Server. Based on the request coming from the request objects, this servlet calls and executes the associated web services to provide the required service.

Client part for the Collaborative Calendar-Server has been integrated into GlobalMMCS Portal. We have the user interfaces implemented as jsp, and JavaScript; users can select their options from the navigation menu, and fill out the necessary fields for posting or retrieving the private or collaborative calendar schedule from the Collaborative Calendar-Server by calling associated services.

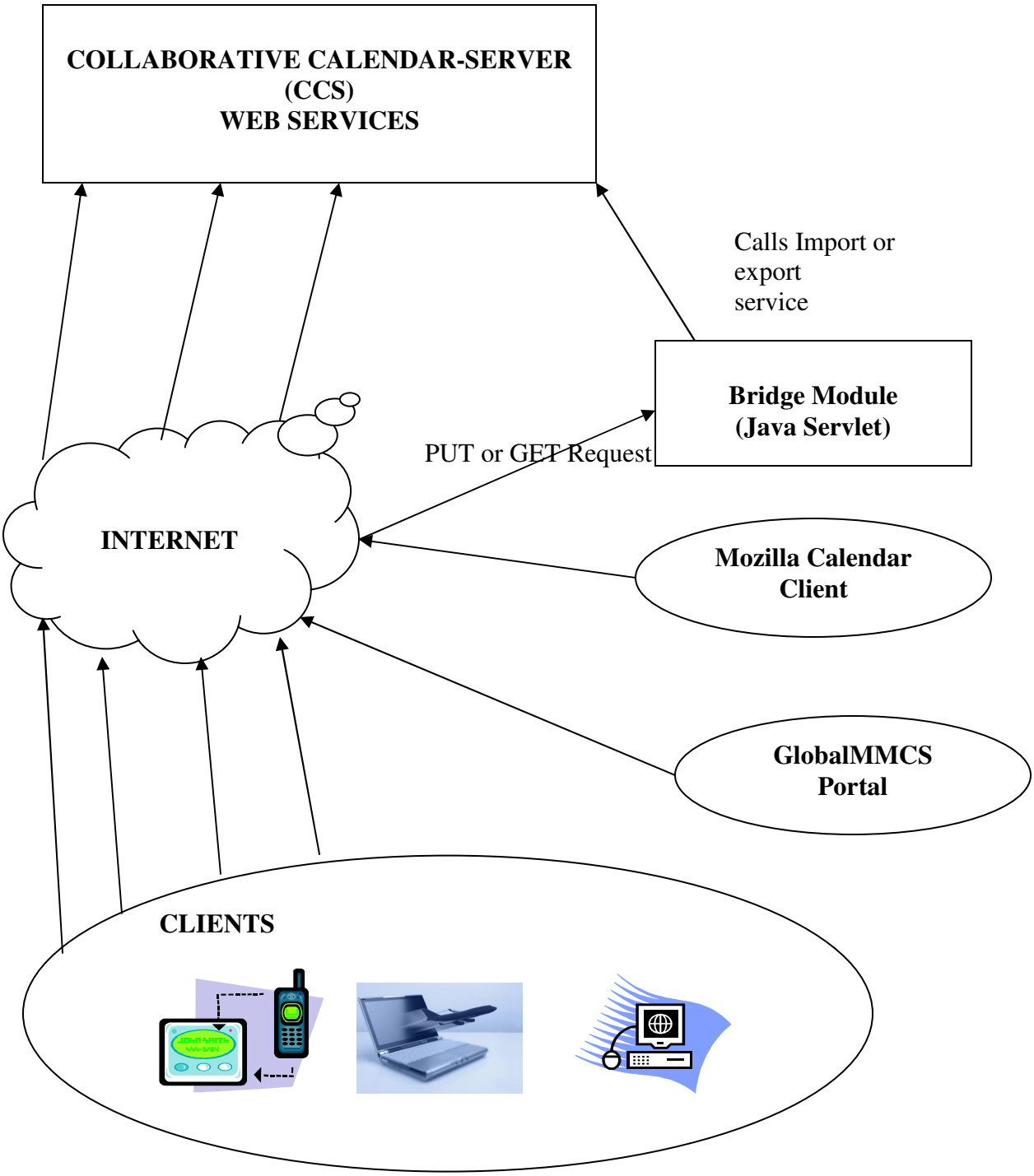


Figure – 1: Collaborative Calendar-Server Architecture

4.1.1 GlobalMMCS Client Module for Collaborative Calendar-Server (CCS)

Clients can invoke Collaborative Calendar-Server's Services in two ways either from a user interface such as a collaborative calendar module of GlobalMMCS Portal implemented in by using jsp and JavaScript or from an application by just calling the associated web service with the required parameters.

After users signed-in to the GlobalMMCS Portal, there are four operations currently supported from the GlobalMMCS Portal's user interface:

- *Private Calendar*: It basically calls the associated services of Collaborative Calendar-Server to retrieve the user's own/private calendar file from the server. When the web service is called, it first checks to see whether this user has an iCalendar file or not on the calendar server. If not, then one empty calendar file is created by using "*makeEmtyCalendar*" service and returned to the user in the html table form. If the user has an iCalendar file, then the required information is read from the calendar file, then html table is constructed, and then it is transferred over http as a text file, and finally showed to the user in the html table form.
- *Collaborative Calendar (Group Calendar)*: This menu option calls the associated CCS's Service to bring up the group/collaborative calendar from the calendar server. Once this service is called, if there is no collaborative calendar file, first it creates an empty calendar file by calling "*makeEmtyCalendar*" service. Then, return an empty html table to the user to show in the client portal. If the collaborative calendar file exists on the server, then the service reads the collaborative calendar file into java object, then setup the html table to be returned to the client.
- *Schedule A Meeting*: This menu item enables users to schedule a meeting into the collaborative calendar. It requires users to fill out the necessary information through the jsp page implemented in the GlobalMMCS portal, and then the associated CCS Service is called to process this request and data. Once the service received this data, it first gets all the GlobalMMCS Portal's registered users. Then, the service checks each the user's calendar file to see whether there is a conflict or not with the new meeting time. If there is no conflict, then the service inserts the new schedule into the collaborative calendar, updates the collaborative calendar file with the latest one, and finally returns a confirmation to the client. If there is any conflict, then it returns a warning to the client so that the client can change the time for this meeting. Time Zone information is calculated based on the client's running location.
- New Event: By this option, users can specify new events into their own/private calendar. Users need to fill out the required fields through the

jsp page implemented in the GlobalMMCS portal such as Event Start Time, Event End Time, Public Event or Private Event, Event Name, Event Location, Event Description, and then the CCS Service is called to process this request and data. Users can specify whether this event is a public or a private event when they are posting it into their private calendar. Once the service receives the data, it first checks to see whether this user has a private calendar or not. If there is not, then it calls the "*makeEmtyCalendar*" service to create one for the client. If there is, then the service reads the user's calendar into the java object, and then adds this new event into his/her private calendar. Finally, the service updates the user calendar file with the latest one. Time Zone information is calculated based on the client's running location.

4.1.2 Collaborative Calendar-Server (CCS)

Collaborative Calendar-Server (CCS) is implemented using java language as collection of Web Services. The CCS's currently implemented services can be reached at: http://gf8.ucs.indiana.edu:18086/CalendarService/servlet/AxisServlet, and can be listed as follow:

- *importCalendar*: This service receives three parameters; icalendar file as a byte array, username as a String, and calendar name as a String. And, it returns a confirmation to the user, if there is any exception, and then the warning is returned to the user as String as well. The service writes the user's calendar file as an iCalendar (calendarname.ics) file under this user's calendar path.

CCS Required Parameters	**Description**	**Example Value**
iCalendar File as a byte[]	It is the iCalendar source file	myIcal[]
Username	Client's username	guest
Calendarname	A name for this calendar	guest

Table-1: The parameters of the importCalendar Service

- *exportCalendar*: The service requires two parameters; username and a calendar name to be exported. It reads the user's calendar file into the byte array, and then returns the iCalendar file as a byte array to the user. If there is any exceptions occur, then it writes those exceptions into a file, and returns it to the user as a byte array as well.

CCS Required Parameters	Description	Example Value
Username	Client's username	guest
Calendarname	A name for this calendar	guest

Table-2: The parameters of the exportCalendar Service

- *makeEmptyCalendar*: It makes an empty calendar for the specified user and for the specified calendar name. It returns a confirmation to the user as a String. If any exceptions occur, then it returns a warning to the client about it.

CCS Required Parameters	Description	Example Value
Username	Client's username	guest
Calendarname	A name for this calendar	guest

Table-3: The parameters of the makeEmptyCalendar Service

- *newEvent*: This CCS service is used for setting up a new event for the user. It basically makes an event for this user, and adds this event into the user's icalendar file as an icalendar VEVENT Component. Finally, updates the user's calendar file with the latest version of it. This service receives the following parameters; username as a String, calendar name as a String, event type (outdoor, private, meeting, holiday etc.) as a String, public or private event as a String, event start time as a long value, time zone id for the start time as a String, event end time as a long value, time zone id for the end time as a String, location of the event as a String, and notes/summary about this event as a String.

CCS Required Parameters	Description	Example Value
Username	Client's username	guest
Calendarname	A name for this calendar file	guest
Event type	Type of the event	Outdoor, meeting, camping etc.
Public/Private Event	Is it a public or a private event	Public (everyone can see)
Start Time	Starting time (long value,milliseconds) for this event.	20051119T135500
Start Time Zone ID	TimeZone ID for start time	America/Indiana/Indianapolis

End Time	Ending time (long value,milliseconds) for this event.	20051119T145500
End Time Zone ID	TimeZone ID for end time	America/Indiana/Indianapolis
Location	Location of this event	Public Library
Notes/Summary	Notes about this event.	Informal discussion meeting.

Table-4: The parameters of the newEvent Service

- *getUserCalendar*: CCS's "getUserCalendar" service requires two parameters; username as a String and a calendar name as a String. When it receives the request from the client, it first locates the user's calendar file on the server, and reads the calendar into java object. Next, it constructs the html table by using the user's calendar data. Finally, it returns the result to the user as a String in html table form. If any exceptions occur, then it returns the exception as a String to the client in html table form.

CCS Required Parameters	**Description**	**Example Value**
Username	Client's username	guest
Calendarname	A name for this calendar	guest

Table-5: The parameters of the getUserCalendar Service

- *scheduleMeeting*: This CCS's service is used for scheduling a meeting into the collaborative/group calendar. Once the request made, the service checks each user's calendar file to retrieve their schedule information, and compare them with the new event time to see if there is any conflict. If there is any conflict with any of the event, then it returns a warning as a String in html table form to the client so that he/she can select another time period for this event. If not, then it sets up a VEVENT, locks the calendar file so that nobody make changes on it concurrently, and adds this event into the collaborative calendar file. Finally, updates the collaborative calendar file on the server with the latest version of it. This service receives the following parameters; username as a String, calendar name as a String, event type (outdoor, private, meeting, holiday etc.) as a String, user names as an String array, event start time as a long value, time zone id for the start time as a String, event end time as a long value, time zone id for the end time as a String, location of the event as a String, and summary about this event as a String.

CCS Required Parameters	Description	Example Value
Username	Client's username	guest
Calendarname	A name for this calendar file	guest
Event type	Type of the event	Outdoor, meeting, camping etc.
User names in String[]	It has the user names to whom schedules be checked with new event schedule	guest, amustaco etc.
Start Time	Starting time (long value,milliseconds) for this event.	20051119T135500
Start Time Zone ID	TimeZone ID for start time	America/Indiana/Indianapolis
End Time	Ending time (long value,milliseconds) for this event.	20051119T145500
End Time Zone ID	TimeZone ID for end time	America/Indiana/Indianapolis
Location	Location of this event	Public Library
Notes/Summary	Notes about this event.	Informal discussion meeting.

Table-6: The parameters of the scheduleMeeting Service

- *getPublicCalendar*: This service requires two parameters in order to return the collaborative/group calendar schedule to the client; username as a String, and a calendar name as a String. When the service receives the request, it first reads the user's calendar file into java object, and then goes through the each component of the calendar (VEVENT, VTODO, VTASK etc) and its properties to construct the schedule to return to the client as a String in html table form. If there is no collaborative calendar defined on the server, then the service creates one by calling "makeEmptyCalendar" web service, and returns the calendar schedule (empty schedule) to the client as a String in html table form as well. If any exceptions occur, then the service returns a warning to the client as a String in html table form.

CCS Required Parameters	Description	Example Value
Username	Client's username	guest
Calendarname	A name for this calendar	guest

Table-7: The parameters of the getPublicCalendar Service

4.1.3 Bridge Module for Collaborative Calendar-Server (CCS)

Our collaborative Calendar-Server can communicate with different calendar clients, which use http methods for communication, through the bridge module such as Mozilla Calendar. A Mozilla Calendar can publish an event(s) to our Collaborative Calendar-Server (CCS), or it can subscribe any of the calendars that exist on the calendar server. This functionality is implemented as a Java Servlet Technology, and the servlet plays a middle layer between the clients and the Collaborative Calendar-Server (CCS) Services. Since, the servlet basically invokes the services of our calendar server based on the coming requests from the clients. For example, if it receives a http "PUT" request, then it calls the importCalendar service of Collaborative Calendar-Server (CCS) to import the events from the client into our calendar server. In this case, client need to specify the required parameters, a username and a calendar name as a String, for the importCalendar service in order to execute the service.

Sample request from Mozilla client to subscribe a calendar on our Collaborative Calendar-Server:

http://gf8.ucs.indiana.edu:28088/CalendarServer/calendar?username=guest&calendarname=guest

4.2 Applicability of Web Service Architecture on to Collaborative Calendaring using iCalendar standards

We have implemented seven operations for Collaborative Calendar-Server as Web Services. These services are importCalendar service, exportCalendar service, getUserCalendar service, makeEmptyCalendar service, newEvent service, scheduleMeeting service, and getPublicCalendar service.

Web Services enables the interoperability between different software applications running on different platforms. Web Services have an interface which is described in a machine-processable format, and web services support interoperable machine to machine interaction over a network. Web Services are defined in a language called Web Services Description Language (WSDL) [13]. The clients can

communicate with a web service by exchanging messages in SOAP (Simple Object Access Protocol) format.

SOAP [14] is a platform and language independent communication protocol for exchanging information in distributed environment. SOAP is an XML based protocol, and consists of three parts the envelope, the encoding rules, and the Remote Procedure Call (RPC) convention. SOAP can be used in any combination of with some other protocols such as HTTP, FTP etc. In our implementation, Collaborative Calendar-Server' Web Services use SOAP over HTTP.

WSDL is specified in XML, and it is used for describing and locating Web Services. WSDL uses four major elements to define Web Services:

- portType: The operations performed by the web service.
- message: Defines the data elements of an operation.
- types: The data types used by the web service.
- binding: Specifies concrete protocol and data format specifications for the operations and messages defined by a particular portType.

We have used Apache Axis version 1.2 to create and publish our Web Services in Collaborative Calendar-Server, and Apache Axis is a reliable and a stable base on which to implement Java Web Services. Furthermore, SOAP communication between client and server is taken care of by Apache Axis.
Using Web Services for implementing Collaborative Calendar-Server Services will offer several key benefits, including:

Integration: It will be easier to integrate Collaborative Calendar-Server functionalities and data into custom applications for developers.

Easy to Extend: Web Service technology is XML based, and Collaborative Calendar-Server is implemented by using Web Services, then it will be easy to extend and configure of the settings to make our implementation more robust and fault tolerant.

Distribution: By using Web Services, it will be easier to spread the calendar data and functionality of services across platforms, operating systems, etc.

5 Conclusion

With the development of Internet Calendaring and Scheduling application and network technique, the calendar data between different applications need to be shared and to be interoperated. iCalendar standard defined in RFC2445 provides the definition of a common format for openly exchanging calendaring and scheduling information across the Internet, and provides the interoperability between the different calendaring and scheduling applications.

Web Service technology provides the interoperable capability of cross-platforms and cross-language in distributed computing environment. Moreover, web service technology is not object oriented and, it overcomes the shortcoming of traditional Distributed Object technique.

In this document, we basically have been trying to explain the efforts spent on building Collaborative Calendar-Server services that supports iCalendar standard. By using web service technology in our implementation, we will take advantage of the Web Services. As Web Services technologies evolve our proposed Collaborative Calendar-Server (CCS) system evolve. WSDL for Collaborative Calendar-Server's services, snapshots of our Collaborative Calendar-Server (CCS) communication with Mozilla Calendar Client and GlobalMMCS client can be found in the appendix part of this document.

APPENDIXES

APPENDIX - 1

Web Service Description File (CalendarServer.wsdl)

```
<?xml version="1.0" encoding="UTF-8" ?>
- <wsdl:definitions targetNamespace="http://webcalendar.webdav.cgl"
    xmlns:apachesoap="http://xml.apache.org/xml-soap"
    xmlns:impl="http://webcalendar.webdav.cgl" xmlns:intf="http://webcalendar.webdav.cgl"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      - <!--
      WSDL created by Apache Axis version: 1.2beta3
      Built on Aug 01, 2004 (05:59:22 PDT)
    -->
- <wsdl:types>
- <schema targetNamespace="http://webcalendar.webdav.cgl"
    xmlns="http://www.w3.org/2001/XMLSchema">
  <import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
- <complexType name="ArrayOfbyte">
- <complexContent>
- <restriction base="soapenc:Array">
  <attribute ref="soapenc:arrayType" wsdl:arrayType="xsd:byte[]" />
    </restriction>
    </complexContent>
    </complexType>
- <complexType name="ArrayOfString">
- <complexContent>
- <restriction base="soapenc:Array">
  <attribute ref="soapenc:arrayType" wsdl:arrayType="soapenc:string[]" />
    </restriction>
    </complexContent>
    </complexType>
    </schema>
    </wsdl:types>
- <wsdl:message name="getUserCalendarResponse">
  <wsdl:part name="getUserCalendarReturn" type="soapenc:string" />
    </wsdl:message>
- <wsdl:message name="importCalendarRequest">
  <wsdl:part name="fileSource" type="impl:ArrayOfbyte" />
  <wsdl:part name="username" type="soapenc:string" />
  <wsdl:part name="calendarName" type="soapenc:string" />
    </wsdl:message>
- <wsdl:message name="getUserCalendarRequest">
  <wsdl:part name="username" type="soapenc:string" />
  <wsdl:part name="calendarname" type="soapenc:string" />
    </wsdl:message>
```

```
- <wsdl:message name="newEventRequest">
  <wsdl:part name="username" type="soapenc:string" />
  <wsdl:part name="calendarName" type="soapenc:string" />
  <wsdl:part name="eventType" type="soapenc:string" />
  <wsdl:part name="public_private" type="soapenc:string" />
  <wsdl:part name="startDate" type="xsd:long" />
  <wsdl:part name="sTimeZoneSt" type="soapenc:string" />
  <wsdl:part name="endDate" type="xsd:long" />
  <wsdl:part name="sTimeZoneEnd" type="soapenc:string" />
  <wsdl:part name="sLocation" type="soapenc:string" />
  <wsdl:part name="notes" type="soapenc:string" />
    </wsdl:message>
- <wsdl:message name="scheduleMeetingResponse">
  <wsdl:part name="scheduleMeetingReturn" type="soapenc:string" />
    </wsdl:message>
- <wsdl:message name="getPublicCalendarRequest">
  <wsdl:part name="username" type="soapenc:string" />
  <wsdl:part name="calendarname" type="soapenc:string" />
    </wsdl:message>
- <wsdl:message name="exportCalendarRequest">
  <wsdl:part name="username" type="soapenc:string" />
  <wsdl:part name="calendarName" type="soapenc:string" />
    </wsdl:message>
- <wsdl:message name="scheduleMeetingRequest">
  <wsdl:part name="username" type="soapenc:string" />
  <wsdl:part name="calendarname" type="soapenc:string" />
  <wsdl:part name="eventType" type="soapenc:string" />
  <wsdl:part name="users" type="impl:ArrayOfString" />
  <wsdl:part name="starttime" type="xsd:long" />
  <wsdl:part name="startTimezone" type="soapenc:string" />
  <wsdl:part name="endtime" type="xsd:long" />
  <wsdl:part name="endTimezone" type="soapenc:string" />
  <wsdl:part name="location" type="soapenc:string" />
  <wsdl:part name="summary" type="soapenc:string" />
    </wsdl:message>
- <wsdl:message name="makeEmptyCalendarResponse">
  <wsdl:part name="makeEmptyCalendarReturn" type="soapenc:string" />
    </wsdl:message>
- <wsdl:message name="importCalendarResponse">
  <wsdl:part name="importCalendarReturn" type="soapenc:string" />
    </wsdl:message>
- <wsdl:message name="makeEmptyCalendarRequest">
  <wsdl:part name="username" type="soapenc:string" />
  <wsdl:part name="calendarName" type="soapenc:string" />
    </wsdl:message>
- <wsdl:message name="exportCalendarResponse">
  <wsdl:part name="exportCalendarReturn" type="impl:ArrayOfbyte" />
    </wsdl:message>
- <wsdl:message name="newEventResponse">
  <wsdl:part name="newEventReturn" type="soapenc:string" />
    </wsdl:message>
```

```
- <wsdl:message name="getPublicCalendarResponse">
  <wsdl:part name="getPublicCalendarReturn" type="soapenc:string" />
    </wsdl:message>
- <wsdl:portType name="CalendarServer">
- <wsdl:operation name="importCalendar" parameterOrder="fileSource username calendarName">
  <wsdl:input message="impl:importCalendarRequest" name="importCalendarRequest" />
  <wsdl:output message="impl:importCalendarResponse" name="importCalendarResponse" />
    </wsdl:operation>
- <wsdl:operation name="exportCalendar" parameterOrder="username calendarName">
  <wsdl:input message="impl:exportCalendarRequest" name="exportCalendarRequest" />
  <wsdl:output message="impl:exportCalendarResponse" name="exportCalendarResponse" />
    </wsdl:operation>
- <wsdl:operation name="getUserCalendar" parameterOrder="username calendarname">
  <wsdl:input message="impl:getUserCalendarRequest" name="getUserCalendarRequest" />
  <wsdl:output message="impl:getUserCalendarResponse" name="getUserCalendarResponse" />
    </wsdl:operation>
- <wsdl:operation name="makeEmptyCalendar" parameterOrder="username calendarName">
  <wsdl:input message="impl:makeEmptyCalendarRequest" name="makeEmptyCalendarRequest"
      />
  <wsdl:output message="impl:makeEmptyCalendarResponse"
      name="makeEmptyCalendarResponse" />
    </wsdl:operation>
- <wsdl:operation name="newEvent" parameterOrder="username calendarName eventType
      public_private startDate sTimeZoneSt endDate sTimeZoneEnd sLocation notes">
  <wsdl:input message="impl:newEventRequest" name="newEventRequest" />
  <wsdl:output message="impl:newEventResponse" name="newEventResponse" />
    </wsdl:operation>
- <wsdl:operation name="scheduleMeeting" parameterOrder="username calendarname eventType
      users starttime startTimezone endtime endTimezone location summary">
  <wsdl:input message="impl:scheduleMeetingRequest" name="scheduleMeetingRequest" />
  <wsdl:output message="impl:scheduleMeetingResponse" name="scheduleMeetingResponse" />
    </wsdl:operation>
- <wsdl:operation name="getPublicCalendar" parameterOrder="username calendarname">
  <wsdl:input message="impl:getPublicCalendarRequest" name="getPublicCalendarRequest" />
  <wsdl:output message="impl:getPublicCalendarResponse" name="getPublicCalendarResponse" />
    </wsdl:operation>
    </wsdl:portType>
- <wsdl:binding name="CalendarServerSoapBinding" type="impl:CalendarServer">
  <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
- <wsdl:operation name="importCalendar">
  <wsdlsoap:operation soapAction="" />
- <wsdl:input name="importCalendarRequest">
  <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="http://webcalendar.webdav.cgl" use="encoded" />
    </wsdl:input>
- <wsdl:output name="importCalendarResponse">
  <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      namespace="http://webcalendar.webdav.cgl" use="encoded" />
    </wsdl:output>
    </wsdl:operation>
- <wsdl:operation name="exportCalendar">
```

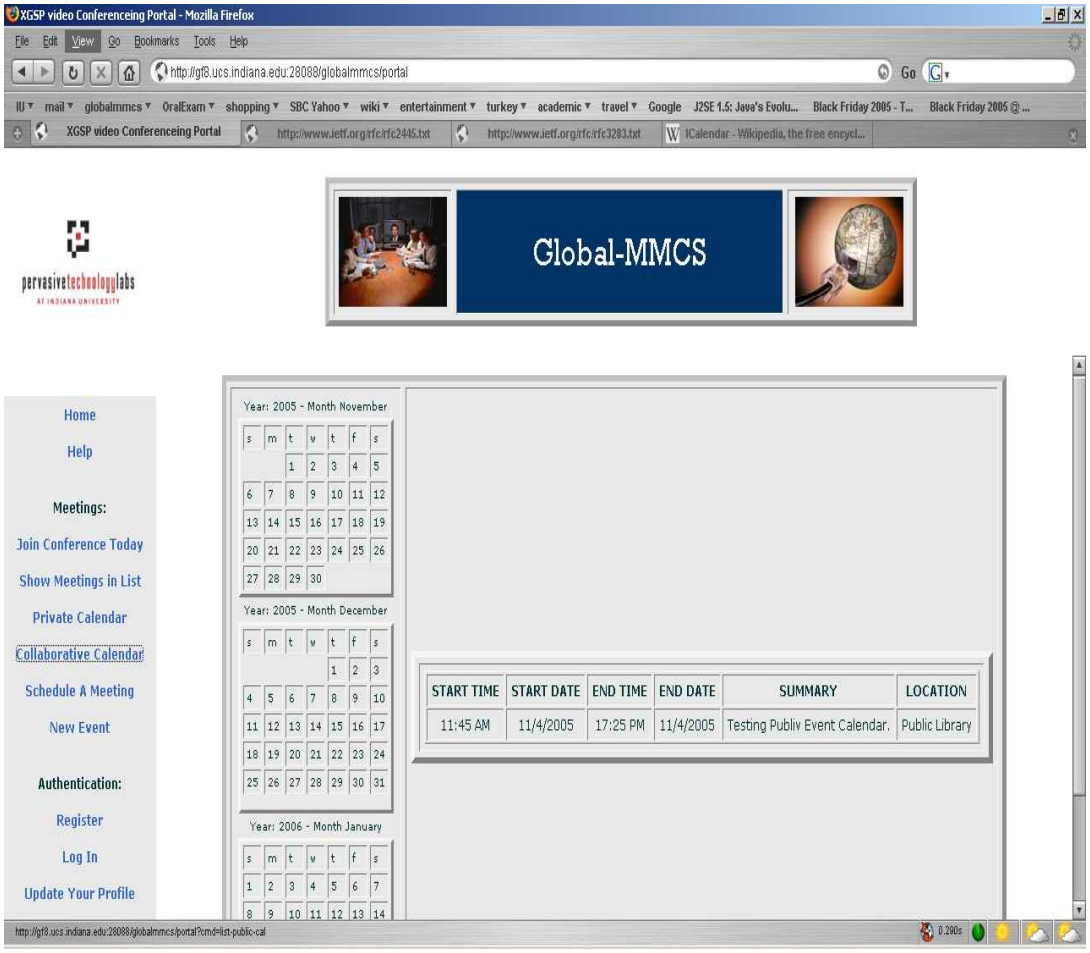
```
  <wsdlsoap:operation soapAction="" />
- <wsdl:input name="exportCalendarRequest">
  <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="http://webcalendar.webdav.cgl" use="encoded" />
      </wsdl:input>
- <wsdl:output name="exportCalendarResponse">
  <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="http://webcalendar.webdav.cgl" use="encoded" />
      </wsdl:output>
      </wsdl:operation>
- <wsdl:operation name="getUserCalendar">
  <wsdlsoap:operation soapAction="" />
- <wsdl:input name="getUserCalendarRequest">
  <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="http://webcalendar.webdav.cgl" use="encoded" />
      </wsdl:input>
- <wsdl:output name="getUserCalendarResponse">
  <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="http://webcalendar.webdav.cgl" use="encoded" />
      </wsdl:output>
      </wsdl:operation>
- <wsdl:operation name="makeEmptyCalendar">
  <wsdlsoap:operation soapAction="" />
- <wsdl:input name="makeEmptyCalendarRequest">
  <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="http://webcalendar.webdav.cgl" use="encoded" />
      </wsdl:input>
- <wsdl:output name="makeEmptyCalendarResponse">
  <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="http://webcalendar.webdav.cgl" use="encoded" />
      </wsdl:output>
      </wsdl:operation>
- <wsdl:operation name="newEvent">
  <wsdlsoap:operation soapAction="" />
- <wsdl:input name="newEventRequest">
  <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="http://webcalendar.webdav.cgl" use="encoded" />
      </wsdl:input>
- <wsdl:output name="newEventResponse">
  <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="http://webcalendar.webdav.cgl" use="encoded" />
      </wsdl:output>
      </wsdl:operation>
- <wsdl:operation name="scheduleMeeting">
  <wsdlsoap:operation soapAction="" />
- <wsdl:input name="scheduleMeetingRequest">
  <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="http://webcalendar.webdav.cgl" use="encoded" />
      </wsdl:input>
- <wsdl:output name="scheduleMeetingResponse">
```

```
  <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="http://webcalendar.webdav.cgl" use="encoded" />
      </wsdl:output>
      </wsdl:operation>
- <wsdl:operation name="getPublicCalendar">
  <wsdlsoap:operation soapAction="" />
- <wsdl:input name="getPublicCalendarRequest">
  <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="http://webcalendar.webdav.cgl" use="encoded" />
      </wsdl:input>
- <wsdl:output name="getPublicCalendarResponse">
  <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="http://webcalendar.webdav.cgl" use="encoded" />
      </wsdl:output>
      </wsdl:operation>
      </wsdl:binding>
- <wsdl:service name="CalendarServerService">
- <wsdl:port binding="impl:CalendarServerSoapBinding" name="CalendarServer">
  <wsdlsoap:address
        location="http://gf8.ucs.indiana.edu:18086/CalendarService/services/CalendarServer" />
      </wsdl:port>
      </wsdl:service>
      </wsdl:definitions>
```

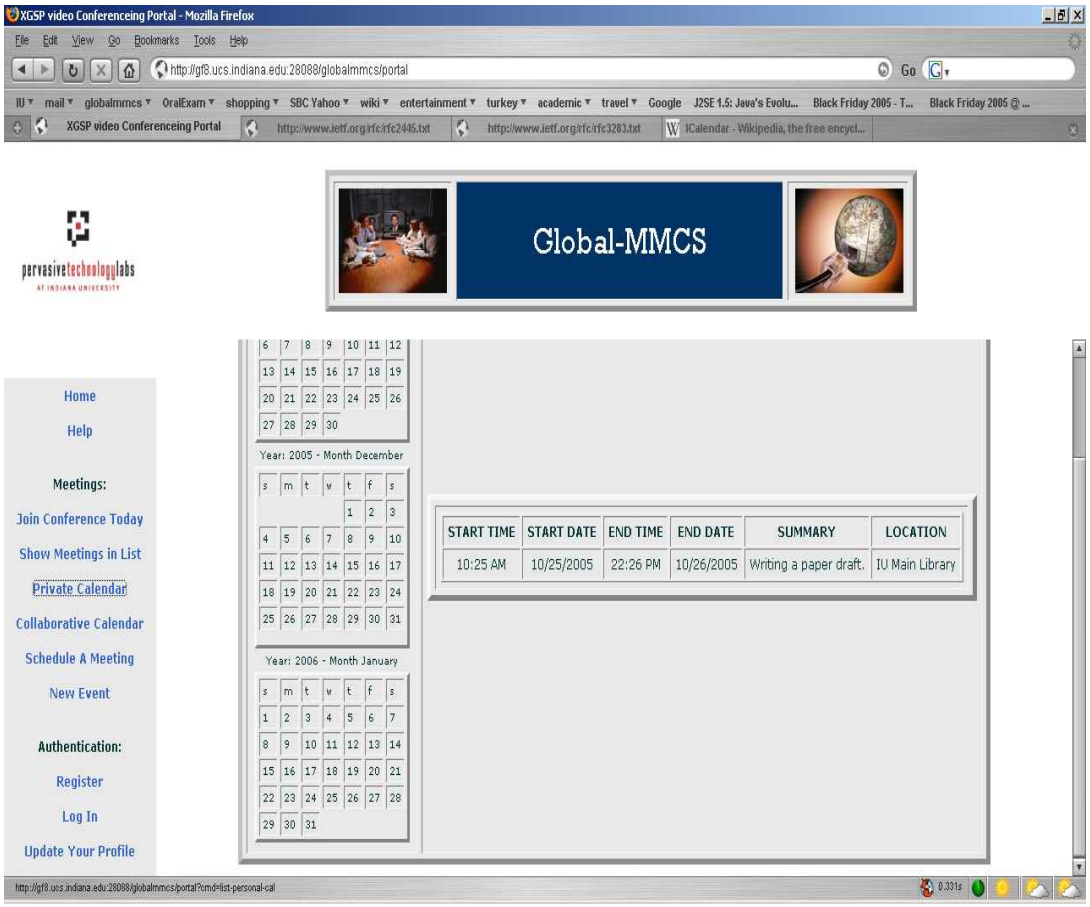
APPENDIX – 2

Client Module deployed on GlobalMMCS Portal for Collaborative Calendar-Server

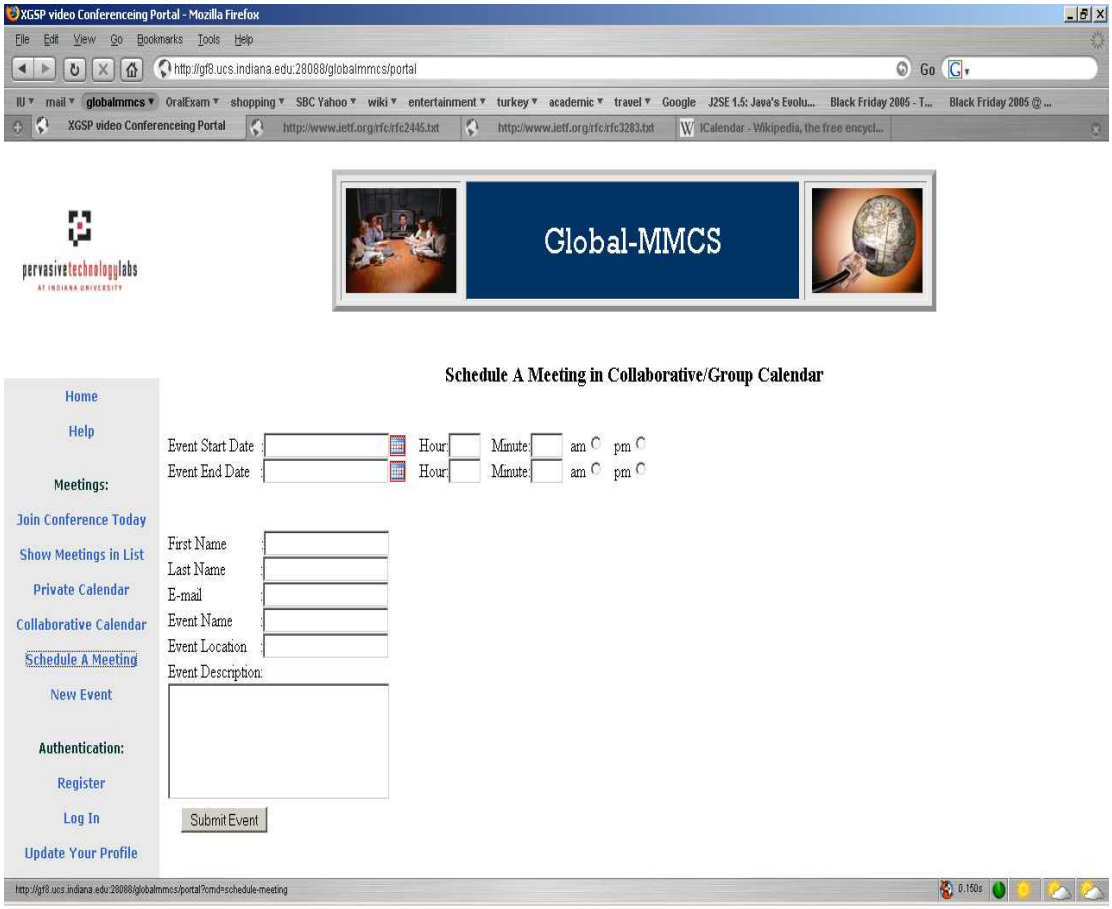
Collaborative Calendar Output:



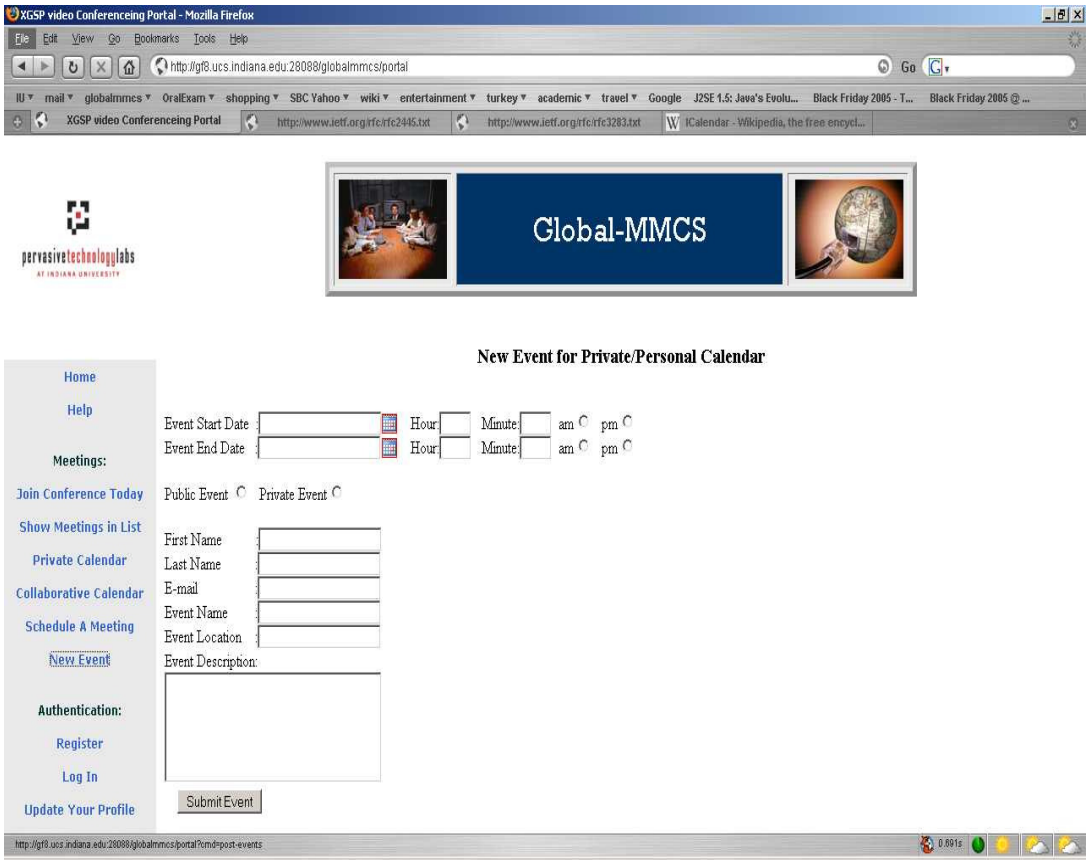
Private Calendar Output:



Scheduling a Meeting in Collaborative/Group Calendar:



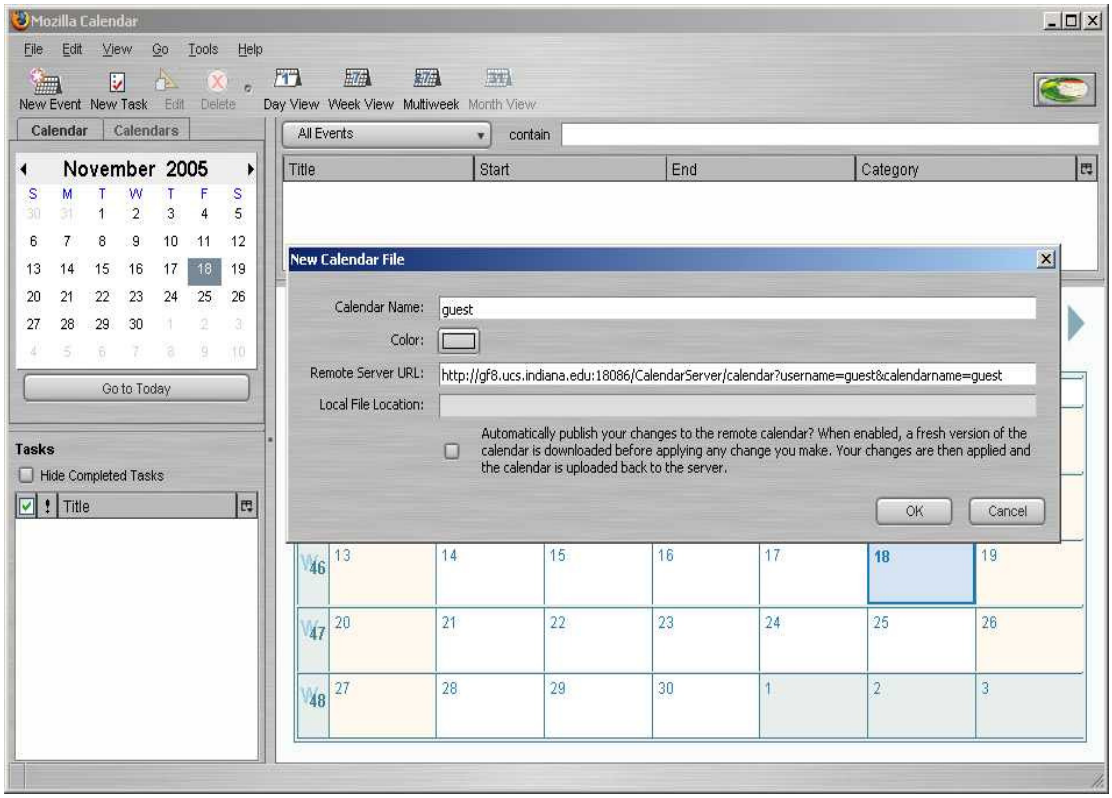
Making a new event into private calendar:



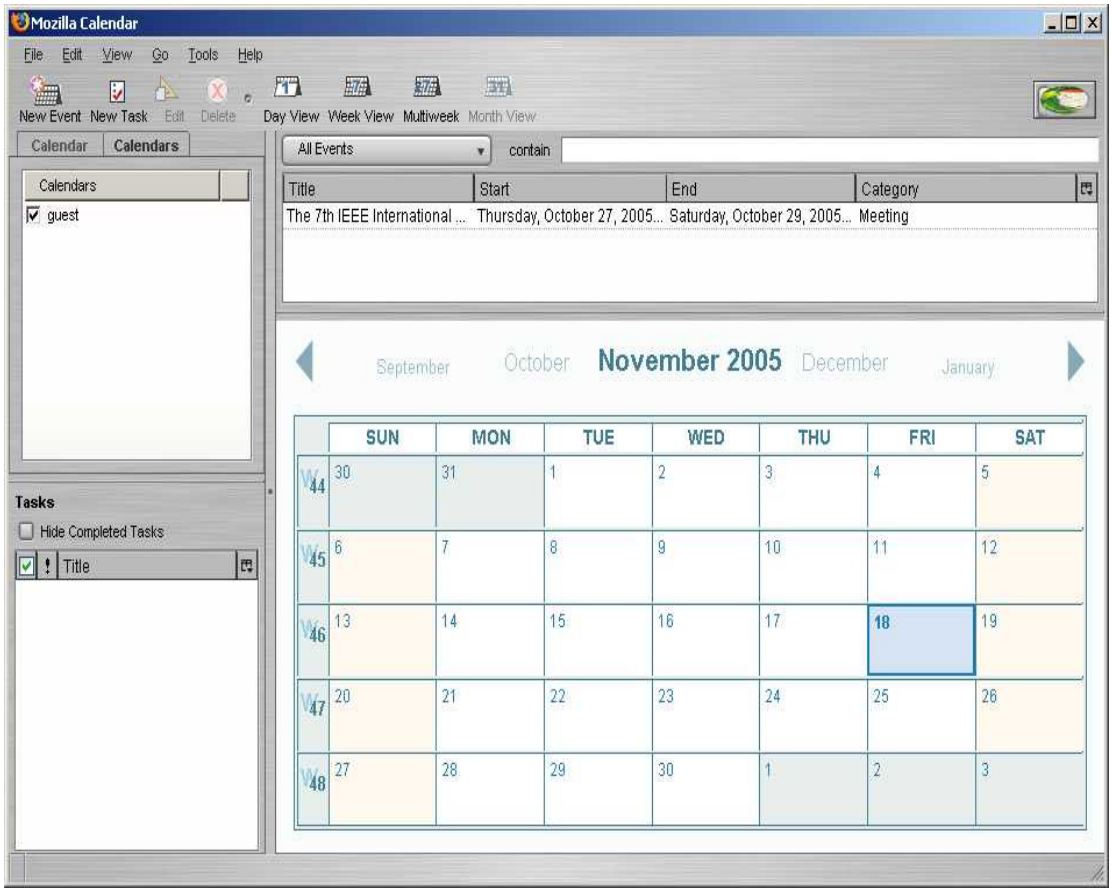
APPENDIX – 3

Collaborative Calendar-Server Communication with Mozilla Calendar Client through the Bridge Module

Subscription request to a calendar through our Collaboration Calendar-Server:



After subscription completed to our calendar server, the calendar can be reached from Mozilla Calendar Client:



REFERENCES

[1] iCalendar (Internet Calendaring and Scheduling Core Object Specification) Standards, official web site http://www.ietf.org/rfc/rfc2445.txt

[2] Schedule World, official web site http://www.scheduleworld.com/

[3] Guide to Internet Calendaring (RFC 3283), official web site http://www.ietf.org/rfc/rfc3283.txt

[4] Kronolith Calendar Application, official web site: http://www.horde.org/kronolith/

[5] The Horde Application Framework, official web site: http://www.horde.org/horde/

[6] Editable shared iCal server, official web site: http://www.afp548.com/Articles/Jaguar/sharedical.html

[7] phpMyCal, official web site: http://dev.neb.net/phpMyCal/

[8] Favorin Time, official web site: http://www.favorin.com/

[9] Chronoss, official web site: http://chronoss.sourceforge.net/

[10] MRBS (Meeting Room Booking System), official web site: http://mrbs.sourceforge.net/

[11] University of Washington Calendar Project, official web site: http://www.washington.edu/ucal/

[12] Scheduling and Calendars, official web site: http://linuxmafia.com/faq/Apps/scheduling.html

[13] Erik Christiensen, Francisco Curbera, Greg Meredith, Sanjiva Weerawarana, Web Service Description Language (WSDL) Version 1.1, March 2001. Available at http://www.w3.org/TR/wsdl

[14] Don Box, David Ehnebuske, Gobal Kakivaya, Andrew Layman, Dave Winer., Simple Object Access Protocol (SOAP) Version 1.1, May 2000. Available at http://www.w3.org/TR/2000/NOTE-SOAP-20000508

[15] iCalendar Wikipedia. Available at: http://en.wikipedia.org/wiki/ICalendar

[16] iCal4j by SourgeFource.net. Available at: http://ical4j.sourceforge.net/

[17] The Internet Engineering Task Force (IETF). Available at: http://www.ietf.org/