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

Ahmet Fatih Mustacoglu[1, 2], Wenjun Wu[1], and Geoffrey Fox[1, 2]
[1]Community Grids Lab, Indiana University, Bloomington, IN, 47404, USA
[2]Department of Computer Science, Indiana University
{amustaco, wewu, gcf}@indiana.edu

ABSTRACT

There has been a clear need to have a common format for representing calendar data to solve the interoperability issues between different types of Calendaring and Scheduling applications. Internet Calendaring and Scheduling Core Object Specification (iCalendar) defines a common format for calendar data exchange to progress the level of interoperability possible between different calendaring and scheduling applications. In this paper, we describe our approach to resolve interoperability issues by providing implementations of a Collaborative Calendar-Server (CCS), a bridge module, and a Global Multimedia Collaboration System (GlobalMMCS) Client Module. The CCS is implemented based on the iCalendar specification as a collection of Web Services. The bridge module allows the CCS to communicate with different Calendaring and Scheduling applications, which are based on the iCalendar specification and use http methods for interactions. GlobalMMCS portal interacts with the CCS to schedule meetings, and to store and to retrieve calendar information through the GlobalMMCS Client Module.

KEYWORDS: iCalendar, Web Services, Calendar Server, Collaborative Calendar, GlobalMMCS Portal.

1. INTRODUCTION

Internet Calendaring and Scheduling Core Object Specification (iCalendar) [1] 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 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) [2] 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.

We need to implement a calendaring and scheduling module for Global Multimedia Collaboration System (GlobalMMCS) Portal [4] of Community Grids Lab at Indiana University so that it can both handle its own calendaring and scheduling needs and also communicate with other scheduling applications. To do so, we have investigated the possible interoperability problems for calendaring and scheduling applications that we might have, and we have decided to handle interoperability issues by implementing the Collaborative Calendar-Server (CCS) based on the iCalendar specification [1]. We have extended the CCS's ability to interact with other calendaring and scheduling applications through the http methods by defining a bridge module, which provides a mechanism to handle coming http requests. Hence, iCalendar based calendaring and scheduling applications can publish and subscribe to a calendar of the CCS. Furthermore, GlobalMMCS Portal users can also access their private calendar and a collaborative calendar of the system, where meeting schedules are made into, by using GlobaMMCS Client module.

This paper gives the details about the design, the architecture, and the implementation details of our Web Service oriented CCS that supports the iCalendar specification. The CCS also has a bridge module that enables other calendaring and scheduling clients to communicate with the CCS such as Mozilla Calendar Client [3], which use http methods to call our services. In this paper, we will first give some background information. Then, we will talk about the design and the architecture of the system. Next, we are going to discuss the CCS's implementation details, which include elaborate explanation of GlobalMMCS [4] Client Module for the CCS, implementation of the CCS as Web Services, and the bridge module for the CCS. Then we are going to present our test results for the CCS. Finally, we will provide the conclusion and future work as a last chapter.

2. BACKGROUND

Grid computing is an emerging technology, and it is the next logical level of distributing computing. Grid is a secure, controlled and coordinated resource sharing among dynamic collections of individuals, institutions, and resources [17]. Interoperability is the central issue in Grid computing. Setting common protocols helps to increase interoperability, and the iCalendar specifications tend to solve interoperability problems among different calendar and scheduling applications. Grid architecture identifies fundamental system components, specifies the purpose and function of these components, and indicates how these components interact with one another. The Open Grid Services Architecture (OGSA) [18], developed by the members of the Global Grid Forum (GGF) [19], defines the grid services and the whole system structure and services that need to be supported in the grid environments. Grid services in OGSA are defined based on Web Services Description Languages (WSDL) with minor extensions. Grid specifications are developing over time, and Web Services Resource Framework (WSRF) is taking the place of Open Grid Services Infrastructure (OGSI), described by OGSA [20]. The main purpose of WSRF is to keep the grid architecture evolution aligned with the developments in Web Services. WSRF is a collection of Web Services specifications that provide a way to access stateful resources using Web Services [21].

The calendaring and scheduling needs have been increasing rapidly for the 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. As a result of this trend, the iCalendar specification have been defined in RFC2445 [1] and intended to advance the interoperability among the calendaring and scheduling applications that are not similar. The iCalendar specification defines the format for specifying iCalendar object methods.

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.) [2], and was authored by Frank Dawson of Lotus Development Corporation [5] and Derik Stenerson of Microsoft Corporation [6]. iCalendar is heavily based on the earlier vCalendar [7] industry specification by the Internet Mail Consortium (IMC) [8].

The CCS has been implemented as Web Services using Java programming language. Web Services leverage the level of interoperability between different software applications that are 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) [14]. The clients can communicate with a web service by exchanging messages in SOAP (Simple Object Access Protocol) format.

SOAP [15] 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 with other protocols such as HTTP, FTP etc. In our implementation, the CCS's Web Services use SOAP over HTTP.

By combining the Web Services technology and the iCalendar specification in the CCS, which can be deployed as a Grid service, we are planning to advance the level of interoperability for different types of applications running on different platforms.

3. DESIGN AND ARCHITECTURE OVERVIEW

The CCS has been designed to provide Web-Service oriented calendaring and scheduling services over the internet. The CCS stores the users' calendar and a collaborative calendar in iCalendar format (with "ics" extension) as specified in RFC 2445 [1] on the machine where the Collaborative Calendar-Server is running. Each user has his/her own iCalendar file for his/her private calendar. There is only one iCalendar file for the collaborative calendar shared by all users. Users' access to the collaborative calendar has been synchronized. An overview of the CCS architecture design is depicted in Figure 1. For the CCS's services please see the project home site [12].

The bridge module, which has been designed as a Java Servlet Technology, provides communication between the CCS and various calendaring and scheduling client applications, which support the iCalendar specification and use http methods to communicate. The CCS can currently interact and communicate with Mozilla Calendar Client [3]. A user, who is using Mozilla Calendar Client [3], can subscribe to a calendar of the CCS, or the user can publish events to a calendar stored on the CCS.

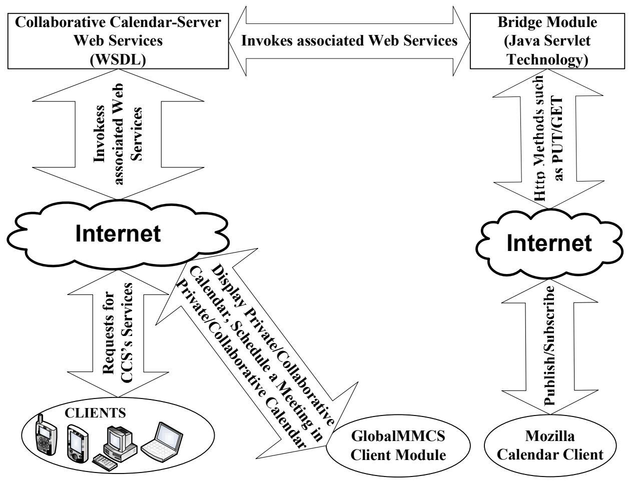
Collaborative Calendar-Server Web Services (WSDL)
Invokes associated Web Services
Bridge Module (Java Servlet Technology)
Invokess associated Web Services
Http Methods such as PUT/GET
Internet
Internet
Requests for CCS's Services
Display Private/Collaborative Calendar, Schedule a Meeting in Private/Collaborative Calendar
Publish/Subscribe
CLIENTS
GlobalMMCS Client Module
Mozilla Calendar Client

Figure 1. Collaborative Calendar-Server Architecture

With CCS's user friendly web-interface integrated into GlobalMMCS portal [10], end users can access their personal calendar (private calendar) or group calendar (collaborative calendar) from anywhere, anytime by using web browser. Users can also schedule a new event into their private calendar or into the group calendar.

4. IMPLEMENTATION

The CCS is implemented as Web Services in pure Java Language using Apache Axis 1.2 [9], and it uses iCal4j Java library [11] supported by SourceForge.net [13] for reading and writing iCalendar data streams as defined in RFC2445 [1]. Java Servlet Technology has been used for the bridge technology, which basically gets the requested http methods and calls the associated Web Service of the CCS based on the coming requests.

We are going to discuss the implementation details of the client module, which is integrated into the GlobalMMCS Portal [10], for the CCS. Next, we are going to give a detailed explanation of web services that composes the CCS. Finally, we are going to express the bridge module implementation of the CCS.

4.1. GlobalMMCS Client Module for CCS

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 the CCS to retrieve the user's private calendar file from the server. When the Web Service is called, it first checks to see whether or not this user has an iCalendar file 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, the required information is read from the calendar file, then the html table is constructed, and 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, it returns an empty html table to the user to show in the client portal. If the collaborative calendar file exists on the server, the service reads the collaborative calendar file into java object, and then it sets up 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 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 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. For the demo please see the GlobalMMCS client implementation home site [16].

4.2. Collaborative Calendar-Server (CCS)

The CCS has been implemented by using java language as a collection of Web Services. The CCS's currently implemented services 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. 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.
- *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 any exceptions occur, then it writes those exceptions into a file, and returns it to the user as a byte array as well.
- *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.
- *New/Add Event*: This service of the CCS 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, it updates the user's calendar file with the latest version. 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.
- *getUserCalendar*: The 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.
- *scheduleMeeting*: This CCS's service is used for scheduling a meeting into the collaborative/group calendar. Once the request is made, the service checks each user's calendar file to retrieve his/her 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 events, 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 makes changes on it concurrently, and adds this event into the collaborative calendar file. Finally, it updates the collaborative calendar file on the server with the latest version. 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 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.
- *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.

4.3. Bridge Module for CCS

The CCS can communicate with different calendar clients, which use http methods for communication, through the bridge module. For example, Mozilla Calendar Client can publish an event(s) to the CCS, or it can subscribe to any of the calendars that exist on the CCS. This functionality is implemented as a Java Servlet Technology, and the Java Servlet plays a middle layer between the clients and the CCS's Services. The Java Servlet basically invokes the services of our calendar server based on the coming requests from the clients. For example, if it receives an http "PUT" request, then it calls the importCalendar service of the CCS to import the events from the client into our calendar server. In this case, the client needs 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 requests can be made from a Mozilla client to subscribe to a calendar on the CCS:
http://gf8.ucs.indiana.edu:28088/CalendarServer/calendar?username=guest&calendarname=guest

5. TEST RESULTS

We performed some tests to evaluate our investigated framework by calculating turnaround time, standard deviation, and standard error values. In each time, our web service has been called 100 times to measure the turn around time, and it has been called 1000 times as a total. Summary of testing environments is depicted in Table 1.

Table 1. Summary of Environment and Machine Configurations

Axis: Running on GridFarm8	
Processor	Intel® Xeon^{TM} CPU (2.40 GHZ)
RAM	2GB total
Bandwidth	100Mbps
OS	GNU/Linux (kernel release 2.4.22)
Java Version	Java 2 platform, Standard Edition(1.5.0_01)
SOAP Engine	AXIS 1.2 and Tomcat 5.0.28

CCS Service Client	
Processor	Intel Centrino Pentium M725 (1.6GHZ)
RAM	512MB total
Bandwidth	100Mbps
OS	Windows XP Professional
Java Version	Java 2 platform, Standard Edition (1.5.0_06)

In Figure 2, we have calculated the average turnaround time for our web services. In each test case, our service returns the requests in an acceptable response time. There is almost no fluctuation for the turnaround time measurement for the CCS Web Services.

We have also measured the Standard Deviation and Standard Error values for the CCS Web Services. Standard Deviation test results are given in Figure 3, and Standard Error test results are given in Figure 4. Our results for Figure 3 and Figure 4 also do not have high fluctuation.

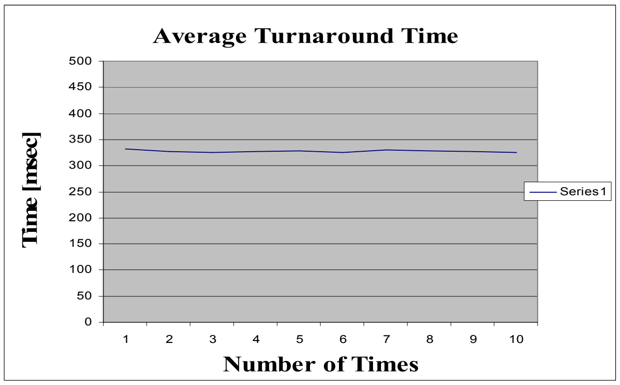

Figure 2. Average Turnaround Time

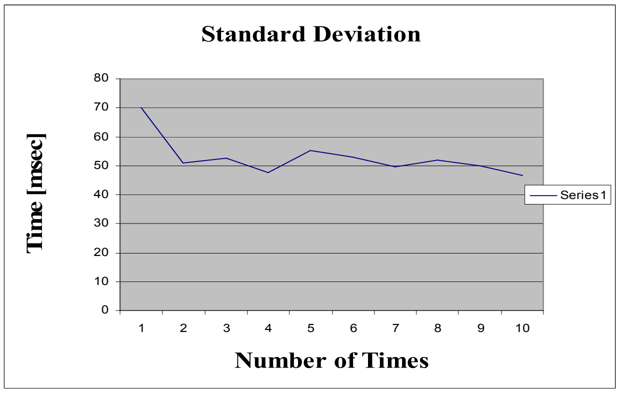

Figure 3. Standard Deviation

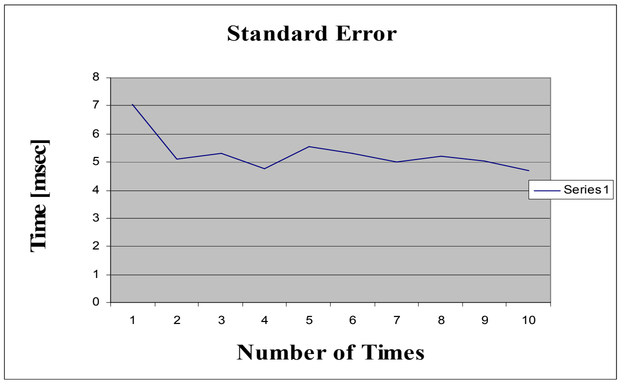

Figure 4. Standard Error

6. CONCLUSION AND FUTURE WORK

The CCS currently can be accessed in various ways such as from GlobalMMCS portal, from any iCalendar based calendaring and scheduling applications or from any application that requests a service from the CCS by using its available Web Services. Password protection has not been implemented in the CCS, since in GlobalMMCS portal users need to have a username and a password in order to access the system. However, for the bridge model of the CCS, we are planning to implement user access list to utilize authorized access into the system. Notification of new event service, which sends a notification to all registered users through the email, will also be implemented in the future.

With the development of Internet Calendaring and Scheduling application and the network technique, the calendar data between different applications need to be shared and to be interoperated. The iCalendar specification 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 Services are essential in grid computing, and Web Service Technology provides the interoperable capability of cross-platforms and cross-language in distributed computing and grid computing environments. Furthermore, 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 the CCS as a collection of Web Services. By using Web Service technology and the iCalendar specification in our implementation, we will improve the level of interoperability between different applications and different platforms. As Web Services technology evolves, our proposed Collaborative Calendar-Server system evolves.

RERERENCES

[1] Internet Calendaring and Scheduling Core Object Specification Web Site: http://www.ietf.org/rfc/rfc2445.txt

[2] The Internet Engineering Task Force Web Site: http://www.ietf.org/

[3] Mozilla Calendar Project Web Site: http://www.mozilla.org/projects/calendar/

[4] Global Multimedia Collaboration System Web Site: http://www.globalmmcs.org/

[5] IBM Lotus Software Web Site: http://www-306.ibm.com/software/lotus/

[6] Microsoft Web Site: http://www.microsoft.com/

[7] Internet Mail Consortium vCard and vCalendar Web Site: http://www.imc.org/pdi/

[8] Internet Mail Consortium Web Site: http://www.imc.org/

[9] Apache AXIS Web Services Web Site: http://ws.apache.org/axis/

[10] Global Multimedia Collaboration System Portal Web Site: http://gf8.ucs.indiana.edu:28088/globalmmcs/portal

[11] ical4j Web Site: http://ical4j.sourceforge.net/

[12] Collaborative Calendar-Server Project web site http://www.opengrids.org/wscalendar/

[13] Sourceforge.net Web Site: http://sourceforge.net/

[14] 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

[15] 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

[16] Global Multimedia Collaboration System Portal Web Site: http://gf8.ucs.indiana.edu:28088/globalmmcs/portal

[17] Ian Foster, Carl Kesselman, Steven Tuecke, The Anatomy of the Grid. Available at: http://www.globus.org/alliance/publications/papers/anatomy.pdf

[18] The Open Grid Services Architecture (OGSA) Web Site: http://www.globus.org/ogsa/

[19] Global Grid Forum (GGF) Web Site: http://www.gridforum.org/

[20] New to Grid Computing by IBM Developer works. Available at: http://www-128.ibm.com/developerworks/grid/newto/

[21]Superceded: Web Services Resource Framework by IBM Developer works. Available at: http://www-128.ibm.com/developerworks/library/specification/ws-resource/