ICalendar Real-time Interoperability Protocol (IRIP)

Status of this Memo

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months. Internet-Drafts may be updated, replaced, or made obsolete by other documents at any time. It is not appropriate to use Internet-Drafts as reference material or to cite them other than as a "working draft" or "work in progress".

To learn the current status of any Internet-Draft, please check the 1id-abstracts.txt listing contained in the Internet-Drafts Shadow Directories on ds.internic.net (US East Coast), nic.nordu.net (Europe), ftp.isi.edu (US West Coast), or munnari.oz.au (Pacific Rim). Distribution of this document is unlimited.

Abstract

This document specifies a binding from the iCalendar Transport-independent Interoperability Protocol [ITIP] to a real-time transport. Calendaring entries defined by the iCalendar Object Model [ICAL] are composed using constructs from [RFC-2045], [RFC-2046], [RFC-2047], [RFC-2048] and [RFC-2049].

This document is based on the calendaring and scheduling model defined by [ICMS].

This document is based on discussions within the Internet Engineering Task Force (IETF) Calendaring and Scheduling (CALSCH) working group. More information about the IETF CALSCH working group activities can be found on the IMC website at http://www.imc.org, the IETF website at http://www.ietf.org/html.charters/calsch-charter.html. Refer to the references within this document for further information on how to access these various documents.

Distribution of this document is unlimited. Comments and suggestions for improvement should be sent to the authors.
1. Introduction

This binding document provides the transport specific information necessary to convey iCalendar Transport-independent Interoperability Protocol [ITIP] over a real-time transport.

1.1 Related Memos

Implementors will need to be familiar with several other memos that, along with this memo, form a framework for Internet calendaring and scheduling standards.

This document specifies an Internet email binding for [ITIP].

[ICMS] - specifies a common terminology and abstract;

[ICAL] - specifies a core specification of objects, data types, properties and property parameters;

[ITIP] - specifies an interoperability protocol for scheduling between different implementations;

[IMIP] - specifies a messaging-based protocol binding for [ITIP].

This memo does not attempt to repeat the specification of concepts or definitions from these other memos. Where possible, references are made to the memo that provides for the specification of these concepts or definitions.

1.2 Formatting Conventions

The mechanisms defined in this memo are defined in propose. In order to refer to elements of the calendaring and scheduling model, core object or interoperability protocol defined in [ICMS], [ICAL] and [ITIP] some formatting conventions have been used.

Calendaring and scheduling roles defined by [ICMS] are referred to in quoted-strings of text with the first character of each word in upper case. For example, "Organizer" refers to a role of a "Calendar User" within the scheduling protocol defined by [ITIP]

Calendar components defined by [ICAL] are referred to with capitalized, quoted-strings of text. All calendar components start with the letter "V". For example, "VEVENT" refers to the event calendar component, "VTODO" refers to the to-do calendar component and "VJOURNAL" refers to the daily journal calendar component.

Scheduling methods defined by [ITIP] are referred to with capitalized, quoted-strings of text. For example, "REQUEST" refers to the method for requesting a scheduling calendar component be created or modified, "REPLY" refers to the method a recipient of a request uses to update their status with the "Organizer" of the calendar component.

Properties defined by [ICAL] are referred to with capitalized, quoted-strings of text, followed by the word "property". For example, "ATTENDEE" property refers to the iCalendar property used to convey the calendar address of a calendar user.

Property parameters defined by [ICAL] are referred to with lower case,
quoted-strings of text, followed by the word "parameter". For example, "VALUE" parameter refers to the iCalendar property parameter used to override the default data type for a property value.

2. Architecture

iRIP enables real-time interoperability between scheduling systems using the iCalendar [ICAL] format for information exchange. iRIP is designed primarily to allow Calendar Services (CS) as defined in [ICSM] to forward real-time requests on behalf of Calendar User Agents (CUA). The goal of iRIP is to allow two or more CS’s to establish connections between them and operate as a single Calendar Domain.

iRIP allows a CS to initiate a session and perform operations on behalf of multiple CUA’s without the need to reauthenticate the session for each CUA.

The design of iRIP does not preclude its use from a CUA directly to a CS. iRIP does not support client access functions such as calendar browsing, retrieval, and search.

Terms used in the following discussion include:

- **User** - the CU that initiates a request.
- **Sender** - the agent used to contact a receiving device, send commands, and receive replies
- **Receiver** - the agent that accepts commands and sends replies.

The Sender and Receiver can take on varying roles of CUA and CS as described in [ICMS]. iRIP allows two CS’s to establish different levels of trust so that scheduling operations can be performed as efficiently as possible. When an iRIP connection is first established, both parties to the connection authenticate one another using the AUTHENTICATE command. The Sender can then initiate commands which the Receiver MUST interpret relative to the Sender’s access control. The AUTHENTICATE command supports proxy operations via [SASL].

2.1 State Diagram

An iRIP session begins when a TCP/IP connection is made on port 5228. The protocol begins in the Connected state. The AUTHENTICATE command, when successful, begins the Authenticated state. From the Authenticated state, the sender can initiate a request using the RECIPIENT command. The Sender can then issues as many RECIPIENT commands as the operation in progress requires until sending an ICALDATA command. After completing the ICALDATA command, the Sender must wait for a response from the receiver. The Receiver can respond that the request has been completed or that the request could not be completed in the time specified by the Sender. When the Receive has ended, the Sender returns to the Authenticated state where another request can be initiated. Implementations should be prepared to handle a DISCONNECT at any point in this state diagram.
2.2 Calendar Address

Calendar addresses are URIs. IRIP uses the following forms of URI.

    [protocol://host.domain[:port]/CSID
    mailto:CSID@host.domain

where:

CSID is an identifier that uniquely identifies the calendar store. There is no implied structure in a CSID, it is an arbitrary string of characters. It may refer to the calendar of a user or of a resource such as a conference room.

protocol is optional. Its default value is "IRIP".

port is optional. Its default value is 5228.

Examples:

    mailto:user1@example.com
    irip://calendar.example.com/user1
    irip://calendar.example.com/conferenceRoomA
    irip://calendar.example.com/89798-098-zytytasd

2.3 Bounded Latency

IRIP is designed so that the Sender can either obtain an immediate response from a request or discover within a known amount of time that the request cannot be completed. When the Sender initiates a command that the Receiver cannot complete within a given amount of time, the
Receiver can return an error code to the Sender indicating this condition. The Sender then issues either a CONTINUE or ABORT command. The ABORT command immediately terminates the command in progress. The CONTINUE command instructs the Receiver to continue processing the command. The ABORT command causes the Receiver to discard the current command and return to the Authenticated state.

3. Commands

In the examples below, lines preceded with "S:" refer to the Sender and lines preceded with "R:" refer to the Receiver.

Reply codes MAY be followed by arbitrary text. The length of the reply code, any subsequent text, and the terminating <CRLF> MUST be 1024 characters or less.

3.1 ABORT

The ABORT command is issued by the Sender to stop an ICALDATA request from being processed further. When the latency time is specified on the ICALDATA command, the Receiver must issue a reply to the Sender within the specified time. The reply may be a reply code indicating that the server has not yet processed the request. The Sender must then tell the server whether to continue or abort.

The ABORT command can also be issued at any time by the Sender after the ICALDATA command has been completed but before the Sender receives a reply.

Example:

...  
S: ICALDATA:10  
R: 2.0.1 Start ICAL input; end with <CRLF>.<CRLF>  
S: Content-Type:text/calendar; method=REQUEST; charset=US-ASCII  
S: Content-Transfer-Encoding: quoted-printable  
S:  
S: BEGIN:VCALENDAR  
S: ...  
S: END:VCALENDAR  
S: .  
<10 seconds elapse...>  
R: .  
R: 7.0  TIMEOUT  
S: ABORT  
R: 2.0  OK  
S: <sender can now begin another command or it can disconnect>

3.2 AUTHENTICATE

The authentication mechanism used in iRIP is based on [SASL]. This allows the iRIP senders and receivers to dynamically negotiate authentication and encryption mechanisms. SASL defines authentication methods such as ANONYMOUS and encapsulates concepts of PROXY used in iRIP.
The AUTHENTICATE command is used by the client to identify itself to the server. Authenticate is required before any other command can be used and must be the first one sent following the server’s welcome message. The format of the command is of the following:

```
AUTHENTICATE <mechanism> <initial data>
```

Example of an authentication session with kerberose version 4:

```
R: 2.0 Welcome IRIP Server
S: AUTHENTICATE KERBEROS_V4 744RTU3r#
S: sfdkgs;ldfjg s;ldfkj gslkfdjgwrt949sl4ns.dlnsgdf
S: sldfljgdsf
gS: slkfjgsdlfjgdsfg
S: ;lasfgsdfg 45243 z!$14325dc
R: 2.0 OK Kerberos V4 authentication successful
```

3.2.1 Authentication with Proxy Access

The proxy mechanism is the ability to have data posted from through an indirect source. To handle this requirement, SASL mechanisms have a separate "Authentication" and "authorization" identity. Thus, server A could authenticate to server B using server A’s credentials with the authorization identity of user X. This effectively allows PROXY operations between servers. Some older SASL mechanisms do not support both authentication and authorization and therefore can’t be used when PROXY operations are required. As per the SASL profile, the authorization identity is the one used to determine if the operation should be allowed or not. The authentication identity ensures the transaction is originating from a trusted sender.

3.2.2 Authentication for Anonymous Access

SASL defines an ANONYMOUS authentication mechanism that must be used if anonymous access is to be implemented by an iRIP capable server. This is done by using the standard SASL authentication method and requesting the ANONYMOUS mechanism. The mechanism consists of a single message from the client to the server. The client sends optional trace information in the for of a human readable string. It is recommended that the trace information take one of three forms: An [RFC-822] Internet e-mail address, an opaque ASCII string which does not contain the "@" character and can be interpreted by the system administrator of the client’s domain or nothing.

The following is an example of anonymous access using an opaque ASCII string:

```
R: <listen on TCP port 5228>
S: <establish a connection to TCP port 5228>
R: 2.0 AboutTime iRIPServer@xyx.com Ready
S: AUTHENTICATE ANONYMOUS
R: +
```
3.2.3 Authentication with plain text password: An example

SASL allows for a server implementation to support simultaneously many different authentication mechanisms, one of them being the well-known plain text password. Although plain text password is a very vulnerable authentication mechanism, it may be adequate for some applications and is illustrated here because it is well known and understood. To properly implement the PLAIN authentication mechanism, refer to [PLAINTRAN-SASL] as listed in the bibliography section of this document.

R: 2.0 Welcome anonymous

3.3 CAPABILITY

The CAPABILITY command tells the server to return a list of capabilities it supports. The server must return a CAPABILITY response with "IRIPrev1" as one of the listed capabilities. The CAPABILITY command can be issued in any connection state and the reply is not dependent upon the connection state.

A capability name which begins with "AUTH=" indicates that the server supports that particular authentication mechanism.

Example:

S: CAPABILITY
R: CAPABILITY IRIPrev1 AUTH=KERBEROS_V4 AUTH=PLAIN
R: 2.0 OK

3.4 CONTINUE

The CONTINUE command is issued by the Sender to allow an ICALDATA request to continue being processed. When the latency time is specified on the ICALDATA command, the Receiver must issue a reply to the Sender within the specified time. The reply may be a reply code indicating that the server has not yet processed the request. The Sender must then tell the server whether to continue or abort.

The CONTINUE has the following form:

    CONTINUE[[:latencyTime]]

If the optional latencyTime is present, it specifies the maximum number of seconds the client will wait for the next response.

In this example, the Sender requests some sort of response from the server every 10 seconds.
3.5 DISCONNECT

The DISCONNECT command signals the end of communication between the Sender and Receiver.

Example:

S: DISCONNECT
R: 2.1 EXAMPLE.COM IRIP Service closing transmission channel

3.6 ICALDATA

The ICALDATA is used specify the iCalendar Object that is to be delivered to one or more recipients specified in the RECIPIENT command. The format of the command is:

S: ICALDATA[:latencyTime]
S: <MIME encapsulated iCalendar Object>
S: <CRLF>.<CRLF>
R: <MIME encapsulated iCalendar Object >
R: <CRLF>.<CRLF>
R: <reply code>

The optional latencyTime value specifies the maximum number of seconds the Sender can wait for a reply. If it is not present, the client places no time limit on the server for a reply. Following the ICALDATA keyword is the iCalendar Object data. The data is terminated by the special sequence <CRLF>.<CRLF>. The server reply may optionally contain an iCalendar Object, the special sequence <CRLF>.<CRLF> followed by a reply code.

S: ICALDATA
R: 2.0.1 Start ICAL input; end with <CRLF>.<CRLF>
S: Content-Type:text/calendar; method=REQUEST; charset=US-ASCII
S: Content-Transfer-Encoding: 7bit
S:
S: BEGIN:VCALENDAR
3.7 RECIPIENT

The RECIPIENT command is used to identify a recipient of the iCalendar Object. Use multiple RECIPIENT commands to specify multiple recipients. The command format is:

RECIPIENT <calendar address>

3.8 SWITCH

The SWITCH command is used to allow the Sender and Receiver to change roles. Its format is:

SWITCH

The SWITCH command is useful in environments where the firewall of a Sender would not allow the Receiver to initiate a connection. The SWITCH command is issued by the Sender to give the Receiver the opportunity to take the role of the Sender. The Sender must be in the authenticated state before the SWITCH command can be used.

The Receiver must respond in one of the following fashions:

a) send an OK reply and take on the role of Sender
b) send a error reply indicating refusal and retain the role of Receiver

If program-A is currently the Sender and sends the SWITCH command and receives an OK reply then program-A becomes the Receiver. Program-A is then in its initial state and sends a service ready greeting message.

If program-B is currently the Receiver and sends an OK reply in response to a SWITCH command then program-B becomes the Sender. Program-B is then in the initial state as if the transmission channel just opened, and expects to receive a service ready greeting.

3.9 Error Codes

iRIP error codes follow the format defined for Status Replies in [ITIP]. All Status Replies as defined in [ITIP] are valid error codes when returned by an iRIP command.

In addition to those defined in [ITIP], iRIP defines the following error codes:

2.0.1 START-ICALDATA
Start ICAL input; end with <CRLF><CRLF>

2.0.2 REPLY-PENDING
A timeout has occurred. The server is still working on the reply. Use CONTINUE
to continue waiting for the reply or
ABORT to terminate the command.

6.0  AUTHORIZATION FAILED  General authorization failure

6.1  TRANSITION-NEEDED  Indicates the transition from
a legacy database to a more
secure password mechanism, and
reports that the new mechanism
is not usable until "AUTHENTICATE
PLAIN" or a login procedure is used.

6.2  AUTH-TOO-WEAK  Indicates that multiple remote
services are offered, and therefore
there is no practical way to transition
every remote service. So, the mechanism
must not allow users to have different
passwords for different services. The
error code reports that no new plaintext
passwords will be accepted from the user
at a later date. It could also indicate
that the requested mechanism is not
available.

6.3  ENCRYPT-NEEDED  Indicates that the requested mechanism
it to be used in conjunction with a
strong external encryption layer.

7.0  TIMEOUT  The requested operation could not be
completed in the time allotted.
Command execution is suspended pending
a reply to continue or abort.

8.0  GENERAL FAILURE  A failure has occurred in the Receiver
that prevents the operation from
succeeding.

8.1  SERVER TOO BUSY  Sent when a connection cannot be
established because the iRIP Receiver
is too busy.

9.0  INVALID IRIP COMMAND  An unrecognized command was received.

10.0  NOT HERE  The Receiver does not know how to
contact the Calendar Store for the
specified RECIPIENT.

10.1  REFERRAL  Accompanied by an alternate address.
The RECIPIENT specified should be
contacted at the given alternate
address.

4. Security Considerations

The security of iRIP with SASL support is highly dependent on the
mechanism used to authenticate the client and whether or not the
security layer is further negotiated. Without a robust security layer, iRIP transactions are subject to eavesdropping and the integrity of iRIP transactions may be compromised. Since iRIP is designed specifically for real time Internet transactions, it is recommended that implementations use the highest degree of authentication and transmission security possible.

Authentication is fundamental to iRIP. It is the basis for granting and denying access. Without a robust security layer iRIP will be subject to many possible attacks and the full contents of the server itself may be at risk.

4.1 SASL ANONYMOUS Mechanism

Implementing support for the Anonymous SASL significantly increases the vulnerability of the calendar server and its data. Refer to [ANON-SASL] for further information on many threats specific to Anonymous SASL access.

4.2 SASL Profile Definition for the protocol

The implementation of SASL in iRIP requires the server and client to comply to the following profile extension:

AUTHENTICATE command. Full description of the challenge/response definition. Starting octet. Interpretation of the authorization identity passed should be interpreted.

4.3 ITIP Threats

Threat: Spoofing the organizer or attendee.

Solution: The entire protection for spoofing attendees or organizer of a meeting resides in the fact that the connection needs to be authenticated. Spoofing would be possible in the absence of authentication.

Threat: Eavesdropping on the traffic.

Solution: If SASL is used to negotiate with the server a security layer, then traffic is no longer in the clear and eavesdropping will not be restricted.

Threat: Flooding of a calendar

Solution: Implementation of iRIP should limit the size and traffic of transaction from a given source.

Threat: Procedural Alarms

Solution: Implementation of iRIP should remove or disallow procedural alarms before delivery.

4.4 IRIP-Specific Threats
Threat: Flooding of connections

Solution: Connections that have not been authenticated within 3 seconds should be disconnected.

5. Examples

5.1 Unauthenticated Freebusy Request

This example shows an anonymous request for the freebusy time of sman@example.com.

R: <listen on TCP port 5228>
S: <establish a connection to TCP port 5228>
R: 2.0 BIG-Time iRIPServer@example.com Ready
S: AUTHENTICATE ANONYMOUS xyz@unknown.com
R: 2.0 Welcome xyz@example.com
S: RECIPIENT:sman
R: 2.0 OK
S: ICALDATA
R: 2.0.1 Start ICAL input; end with <CRLF>.<CRLF>
S: Content-Type:text/calendar; method=REQUEST; charset=US-ASCII
S: Content-Transfer-Encoding: 7bit
S:
S: BEGIN:VCALENDAR
S: PRODID:-//ACME/DesktopCalendar//EN
S: METHOD:REQUEST
S: VERSION:2.0
S: BEGIN:VFREEBUSY
S: ORGANIZER:mailto:xyz@unknown.com
S: ATTENDEE:mailto:sman@example.com
S: DTSTAMP:19971113T190000Z
S: BEGIN:VFREEBUSY
S: ATTENDEE:mailto:sman@example.com
S: DTSTAMP:19971113T190005Z
S: DTSTART:19971115T160000Z
S: DTEND:19971116T040000Z
S: UID:www.example.com-873970198738777@host.com
S: FREEBUSY:19971113T230000Z/PT1H,19971114T210000Z/PT30M
S: END:VFREEBUSY
S: END:VCALENDAR
S: .

R: Content-Type:text/calendar; method=REPLY; charset=US-ASCII
R: Content-Transfer-Encoding: 7bit
R:
R: BEGIN:VCALENDAR
R: PRODID:-//EXAMPLE/DesktopCalendar//EN
R: METHOD:REPLY
R: VERSION:2.0
R: BEGIN:VFREEBUSY
R: ATTENDEE:mailto:sman@example.com
R: DTSTAMP:19971113T190000Z
R: DTSTART:19971115T160000Z
R: DTEND:19971116T040000Z
R: UID:www.example.com-873970198738777@host.com
R: FREEBUSY:19971113T230000Z/PT1H,19971114T210000Z/PT30M
R: END:VFREEBUSY
R: END:VCALENDAR
R: .
5.2 Using Switch

This session demonstrates how a poll can be accomplished using the SWITCH command. In this case, the sender (S) becomes the receiver (R) after issuing the switch command.

5.3 Resource Scheduling
S: RECIPIENT Large_Conference_Room
R: 2.0 OK
S: RECIPIENT Overhead_Projector_1
R: 2.0 OK
S: ICALDATA: 10
S: BEGIN:VCALENDAR
S: PRODID:-//Foo Corporation//Insight MIMEDIR//EN
S: VERSION:2.0
S: BEGIN:VEVENT
S: DTSTART:19980708T220000Z
S: DTEND:19980708T230000Z
S: LOCATION:Large Conference Room
S: DESCRIPTION:Another big customer meeting in Large Conference room.
S: SUMMARY:Customer meeting
S: PRIORITY:1
S: END:VEVENT
S: END:VCALENDAR
S: .
R: <looks up availability in database>
R: .
R: 4.0 No, Large_Conference_Room not available at that time.
S: RECIPIENT Large_Conference_Room
R: 2.0 OK
S: RECIPIENT Overhead_Projector_1
R: 2.0 OK
S: ICALDATA: 10
S: BEGIN:VCALENDAR
S: PRODID:-//Foo Corporation//Insight MIMEDIR//EN
S: VERSION:2.0
S: BEGIN:VEVENT
S: DTSTART:19980708T160000Z
S: DTEND:19980708T170000Z
S: LOCATION:Large Conference Room
S: DESCRIPTION:Another big customer meeting in Large Conference room.
S: SUMMARY:Customer meeting
S: PRIORITY:1
S: END:VEVENT
S: END:VCALENDAR
S: .
R: <looks up availability in database>

Courtemanche/Mansour/O’Leary       13               Expires February 1998

Internet Draft                     iRIP                     August 3, 1998

R: <10 seconds pass>
R: .
R: 7.0 Database too busy, try again later
S: DISCONNECT
R: 2.0 OK
R: <drops TCP connection>

6. Acknowledgments

The following have participated in the drafting and discussion of this memo:

Doug Royer, Mugino Saeki

7. Bibliography
[ANON-SASL] "Anonymous SASL mechanism", draft-newman-sasl-anon-00.txt

[PLAINTRAN-SASL] "Plain text password SASL mechanism", draft-newman-sasl-plaintrans-03.txt


8. Open Issues

Registration of the SASL profile for iRIP with the IANA.
Calendar addressing
Port Number registration

9. Author’s Address

The following address information is provided in a vCard v2.1, Electronic Business Card, format.

BEGIN:VCARD
FN:Andre Courtemanche
ORG:CS&T
ADR;WORK;POSTAL;PARCEL:;;3333 Graham Boulevard;Montreal;QC;H3R 3L5;Canada
TEL;WORK;MSG:+1-514-733-8500
TEL;WORK;FAX:+1-514-733-8788
EMAIL;INTERNET:andre@cst.ca
END:VCARD

BEGIN:VCARD
FN:Steve Mansour
ORG:Netscape Communications Corporation
ADR;WORK;POSTAL;PARCEL:;;501 East Middlefield Road;Mountain View;CA;94043;USA
TEL;WORK;MSG:+1-650-937-2378
TEL;WORK;FAX:+1-650-937-2103
EMAIL;INTERNET:sman@netscape.com
END:VCARD

BEGIN:VCARD
FN:Pete O’Leary
ORG:Amplitude
ADR;WORK;POSTAL;PARCEL:;;
TEL;WORK;MSG:+1-415-659-3511
TEL;WORK;FAX:+1-415
EMAIL;INTERNET:pete@amplitude.com
END:VCARD

The iCalendar object is a result of the work of the Internet Engineering Task Force Calendaring and scheduling Working Group. The chairman of that working group is:

BEGIN:VCARD
FN:Anik Ganguly
END:VCARD
10. Full Copyright Statement

Copyright (C) The Internet Society (date). All Rights Reserved.

This document and translations of it MAY be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation MAY be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself MAY NOT be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process MUST be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.