Network Working Group   Andre Courtemanche, CS&T
Internet Draft   IRIP   Steve Mansour, Netscape
<draft-ietf-calsch-irip-03.txt>   Pete O'Leary, Amplitude
Expires 6 months after: April 16, 1999

ICalendar Real-time Interoperability Protocol (iRIP)

Status of this Memo

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 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.

Table Of Contents

1 Introduction 3
1.1Related Memos 3
1.2Formatting Conventions 3
2 Architecture 4
2.1Protocol States 5
2.2Calendar Address 6
2.3Bounded Latency 7
3 Protocol 7
3.1Commands 7
3.1.1ABORT 8
3.1.2AUTHENTICATE 9
3.1.3CAPABILITY 12
3.1.4CONTINUE 13
3.1.5DEQUEUE 14
3.1.6DISCONNECT 15
3.1.7RECIPIENT 15
3.1.8SENDATA 17
3.1.9SWITCH 19
3.2Fanout and Queued Transactions 19
3.3Bi-Directional Queue Operation 20
3.4Reply Codes 20
4 Implementation Considerations 23
5 Security Considerations 23
5.1Security Threats and Recommendations 23
5.1.1Authentication Hacking 23
5.1.2Spoofing 23
5.1.3Eavesdropping 24
5.1.4Connection Flooding 24
5.2Security Interoperability Issues 24
6 Examples 24
6.1Unauthenticated Freebusy Request 24
6.2Busy Time Request 25
6.3Using Switch 28
6.4Fanout Requests 29
6.4.1Successful Fanout Request 29
6.4.2Referral On Fanout 30
6.5Queued Requests 31
6.5.1Meeting Invitation 32
7 Acknowledgments 33
8 Bibliography 33
9 Open Issues 34
10 Author's Address 34
11 Full Copyright Statement 36

1 Introduction

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

1.1 Related Memos

Implementers 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 a real-time binding for [ITIP].

- [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 document 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 [ICAL] and [ITIP] some formatting conventions have been used.

Calendaring and scheduling roles defined by [ITIP] 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) to forward real-time requests on behalf of Calendar User Agents (CUA) and receive real-time responses. The goal of iRIP is to allow two or more CS's to establish connections with each other. However, the design of iRIP does not preclude its use from CUA directly to CS. 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 sections and examples below refer to a "user", a "sender", and a "receiver". For purposes of this document these terms are defined as follows:

```
user     - the Calendar User 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.
```

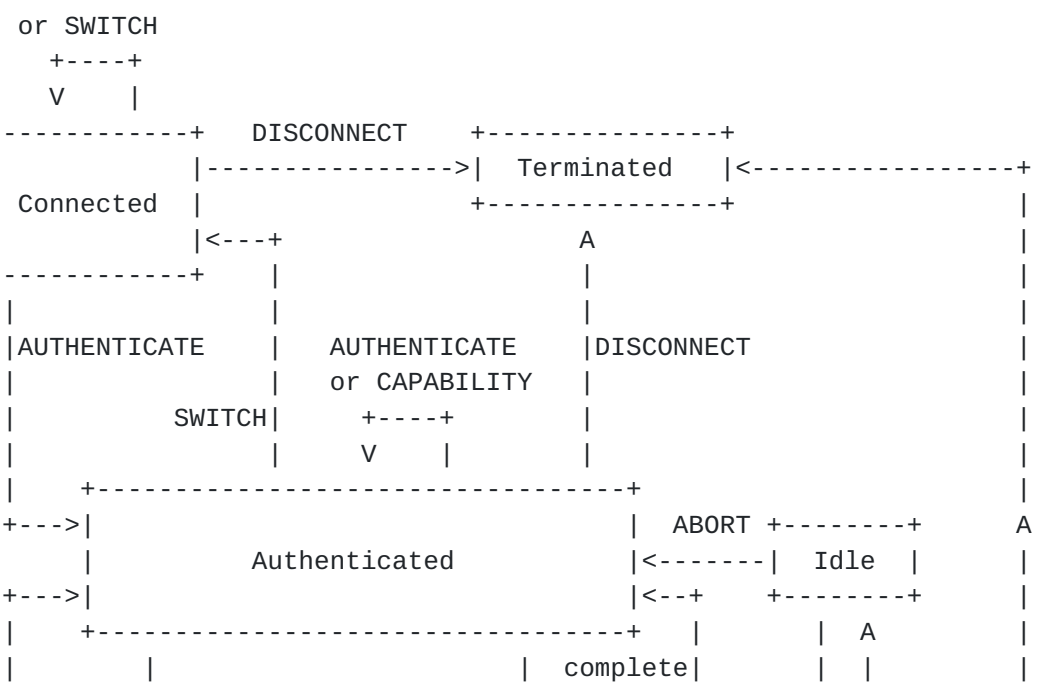
The sender and receiver can take on varying roles of a Calendar User Agent (CUA) and Calendar Store (CS).

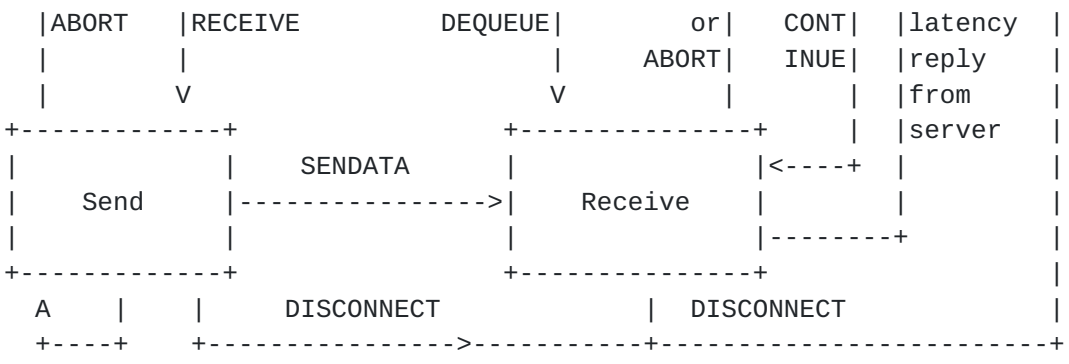
iRIP allows two CS's to establish different levels of trust. When an iRIP connection is first established, the sender CS authenticates as

the iRIP server acting as a proxy for the originator of each ITIP message being sent to the receiver.

2.1 Protocol States1

The iRIP state diagram is shown below. The states are shown in the boxes. State names are written with the first letter capitalized. The commands used to switch between states are shown next to an arrow connecting the states. The commands are listed in all capital letters. A condition that causes a state to change is shown in lower case letters.





An iRIP session begins when a TCP/IP connection is made on port 5228. The protocol begins in the Connected state. Once connected, the sender can issue the CAPABILITY which leaves the protocol in the Connected state. The sender can also issue the SWITCH command requesting the receiver to switch roles with the sender. Whether the receiver accepts or declines the request, the protocol remains in the Connected state. The DISCONNECT command terminates the connection. The AUTHENTICATE command, when successful, begins the Authenticated state. From the Authenticated state, the sender can:

1. begin sending an ITIP message to the receiver by issuing the

RECIPIENT command,

2. **re-authenticate as a different calendar using the AUTHENTICATE command,**
3. **request a queued ITIP message for the authenticated calendar using the DEQUEUE command,**
4. **execute the CAPABILITY command,**
5. **disconnect**

In order to send an ITIP message to other calendars, the sender begins by issuing the RECIPIENT command causing the protocol to enter the Send state. The sender repeats the RECIPIENT command as many times as needed to indicate all the target calendars. When all recipients have been specified, the sender issues the SENDATA command and supplies the [ITIP] message. This causes the protocol to enter the Receive state where the sender waits for a response from the receiver. If the receiver's response indicates that the request has been completed the protocol returns to the Authenticated state. If the receiver indicates that the request could not be completed in the time specified by the sender the protocol enters the Idle state. At this point the sender must decide how to proceed. If the sender issues the CONTINUE command, the command in progress continues and the session returns to the Receive state. If the sender issues the ABORT command the command is

aborted and the session is returned to the Authenticated state.

The sender may decide to abort sending the ITIP message while it issues the RECIPIENT commands (perhaps because a RECIPIENT does not exist). If the sender issues the ABORT command after one or more RECIPIENT commands, the protocol returns to the Authenticated state.

A sender can abort an operation in progress while it is in the Receive state by sending an ABORT command to the receiver.

>From the Authenticated state, a sender can also issue the DEQUEUE command causing the protocol to request the receiver to return a single queued ITIP message. Issuing the DEQUEUE command changes the protocol to the Receive state. The receiver replies with a single queued [ITIP] request or a status code to indicate that there are no more queued requests for the authenticated user.

Though the DISCONNECT command should only be issued from the Authenticated or Connected states, implementations should be prepared to handle a DISCONNECT at any point in this state diagram.

2.2 Calendar Address

Calendar addresses, or CALUIDs, are URIs that are modeled after [RFC2396]. iRIP CALIDs use the following form of URI.

    [<scheme>://<host>[:<port>]/]<relativeCALUID>

where:

    <scheme> must be "irip"

    <host> is address of the computer on which the iRIP server is
           running. This is also called the Calendar Store ID or
           CSID.

    <port> is optional. Its default value is 5228.

    <relativeCALUID> is an identifier that uniquely identifies the
           calendar on a particular calendar store. There is no
           implied structure in a relativeCALUID, it is an arbitrary
           string of 7-bit ASCII characters. It may refer to the
           calendar of a user or of a resource such as a conference
           room. It MUST be unique within the calendar store. It is
           recommended that the relativeCALUID be globally unique.

If the CSID is present the CALID is said to be "qualified". Qualified CALIDs are necessary when the CSID portion of a calendar address is different from the Calendar Store on which the calendar user is currently authenticated.

Examples:

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

For the iRIP server on calendar.example.com, the first two addresses refer to the same calendar.

2.3 Bounded Latency

iRIP is designed so that the sender can either obtain an immediate response from a request or discover within a specified amount of time that the request has not yet completed. The sender can initiate commands with an optional latency time specified. When the sender specifies the latency time and the receiver cannot complete the operation within the specified amount of time, the receiver return an appropriate response code to the sender. 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 Protocol

3.1 Commands

iRIP commands are summarized in the table below and described in detail in the following sections.

Command	Issued from State

ABORT	Idle, Send, Receive
AUTHENTICATE	Connected, Authenticated
CAPABILITY	Connected, Authenticated, Send
CONTINUE	Idle
DEQUEUE	Authenticated
DISCONNECT	Connected, Authenticated
SENDATA	Send
RECIPIENT	Authenticated, Send
SWITCH	Connected, Authenticated

Commands have the general form:

<command> [arguments...]

where <command> is a command listed in the table above. A command MAY have arguments. Arguments are defined in the detailed command definitions below. The length of a command and its arguments, and the terminating <CRLF> MUST be l024 characters or less.

Responses to commands have the following general form:

[ICAL OBJECT]
.
<reply code> [arguments...]

A response MAY include an [ICAL] object. Whether an [ICAL] object is present or not, the sequence <CRLF>.<CRLF> followed by a reply code is mandatory. The reply code may have associated arguments. These arguments are defined in the command descriptions for each command. Arguments MUST NOT be present unless they are specifically called for by the particular reply code.. The length of the reply code, any arguments, and the terminating <CRLF> MUST be l024 characters or less.

In the examples below, lines preceded with "S:" refer to the sender and lines preceded with "R:" refer to the receiver. Lines in which the first non-whitespace character is a "#" are editorial comments and are not part of the protocol.

3.1.1 ABORT

Arguments: none

Data: none

```
Result:     2.0 - success
            2.2 - no command in progress
```

The ABORT command is issued by the sender to stop a command whose latency time has been exceeded. When the latency time is specified on the SENDATA 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 Sender can issue the ABORT command at any time after the SENDATA command has been completed but before the sender receives a reply.

Example:

```
...
S: RECIPIENT irip://cal.example.com/abc
R: 2.0 irip://cal.example.com/abc
S: RECIPIENT def
R: 2.0
S: SENDATA 10
R: 2.0.1
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: 2.0.2 irip://cal.example.com/abc
R: 2.0.2 irip://cal.example.com/def
S: ABORT
R: 2.0.3
S: <sender can now begin another command or it can disconnect>
```

The Receiver will issue the 8.2 reply code if it receives an ABORT when the SENDATA or DEQUEUE command is not in progress. This could happen if the Sender issues an ABORT command at a point in time after the Receiver has completed the operation and issued the reply code but before the Sender has actually received the reply code. For example:

```
S: SENDATA 10
S: <an ICAL object>
S: .
# 10 seconds elapse...
S: ABORT
```

R: 2.0
R: 8.2

In this case, the reply code 2.0 is in response to the [ICAL] object and the reply code 8.2 is in response to the ABORT command.

3.1.2 AUTHENTICATE

Arguments: <SASL mechanism name> [<initial data>]

Data: continuation data may be requested

Result: 2.0 - Authenticate completed, now in authenticated state
6.0 - Failed authentication
6.1 - authenticate failure: unsupported authentication mechanism, credentials rejected
6.2 - Sender aborted authentication, authentication exchange cancelled
6.3 - Unsupported Authentication Mechanism
9.1 - Unexpected command.

The AUTHENTICATE command is used by the client to identify the user to the server. iRIP uses the [SASL] specification for authentication. This allows iRIP users to choose from a variety of authentication mechanisms. The only iRIP commands which can be issued before authentication occurs are AUTHENTICATE, CAPABILITY, SWITCH and DISCONNECT.

The AUTHENTICATE command initiates the authentication protocol exchange.

<SASL mechanism name> is a registered SASL authentication mechanism. (Refer to [SASL] for information on obtaining a list of currently registered mechanisms.) <initial data> is an optional parameter which can be used for mechanisms which require an initial Sender response.

If the mechanism is not supported by the Receiver it must indicate this with a "." CRLF and the reply code 6.1 indicating that the authentication mechanism is not supported. Supported authentication mechanisms can be discovered using the CAPABILITY command.

If the mechanism is supported an authentication protocol exchange takes place, in the form of a series of Receiver challenges and Sender responses. The Receiver terminates the exchange with the <CRLF>.<CRLF> sequence followed by a reply code. Successful authentication is indicated with the reply code 2.0, and unsuccessful authentication is indicated with the reply code 6.0. If the authentication was successful, but the authorization identity was not

accepted the status code 6.3 is used. Upon successful authentication the protocol enters the Authenticated state, otherwise it remains in the Connected state.

In the authentication protocol exchange both Receiver challenges and Sender responses consist of the authentication mechanism data transformed into BASE64 and followed by a CRLF. If the Sender wishes to cancel an authentication exchange, it issues the <CRLF>.<CRLF> sequence. Upon receipt of such an answer, the Receiver MUST indicate the end of the exchange the <CRLF>.<CRLF> sequence followed by reply code 6.2 indicating that the exchange was aborted.

If a security layer was negotiated it comes into effect for the Receiver starting with the first octet transmitted after the CRLF which follows the 2.0 reply code, and for the Sender starting with the first octet after the CRLF that concludes the authentication exchange for the client. Data is transmitted as described in [SASL].

The service name specified by this protocol's profile of SASL is "irip".

The following examples illustrate the various possiblities for an authentication protocol exchange using Kerberos Version 4.

Successful authentication:

```
S: AUTHENTICATE KERBEROS_V4
R: AmFYig==
S: BAcAQU5EUkVXLkNNVS5FRFUAOCAsho84kLN3/IJmrMG+25a4DT
R: or//EoAADZI=
S: DiAF5A4gA+oOIALuBkAAmw==
R: .
R: 2.0
```

Failed authorization:

```
S: AUTHENTICATE KERBEROS_V4
R: AmFYig==
S: BAcAQU5EUkVXLkNNVS5FRFUAOCAsho84kLN3/IJmrMG+25a4DT
R: or//EoAADZI=
S: DiAF5A4gA+oOIALuBkAAmw==
R: .
R: 6.3
```

Failed authentication:

```
S: AUTHENTICATE KERBEROS_V4
R: AmFYig==
S: BAcAQU5EUkVXLkNNVS5FRFUAOCAsho84kLN3/IJmrMG+25a4DT
R: .
R: 6.0

Sender aborted authentication:

S: AUTHENTICATE KERBEROS_V4
```



```
R: AmFYig==
S: .
R: .
R: 6.2

Unsupported mechanism:

S: AUTHENTICATE Experimental_Auth
R: .
R: 6.1
```

3.1.2.1 Authentication with Proxy Access

Some [SASL] mechanisms allow the Sender to transmit an authorization identity which is different from the authentication identity. iRIP depends upon this ability in that it is servers who authenticate to each other in order to process requests for users. The user which the Sender is representing is transmitted as the authorization identity during the [SASL] exchange. This identity takes the form of a CALID.

The authorization identity is used to administer the [ITIP] security paradigm. Thus in an iRIP session REQUESTs may be issued for events of which the authorized CALID is the Organizer, RESPONSEs or COUNTERs may be issued for events which the authorized caladdress is Attending, etc.

3.1.2.2 Selection of an Authentication Mechanism

The authentication mechanisms which a Receiver supports may be discovered through use of the CAPABILITY. From the supplied list the Sender may choose its preferred mechanism.

Not all mechanisms will be supported on all servers. There is no minimum level of security which an iRIP compliant server is required to support. This may result in iRIP servers which are unable to talk

to each other through lack of a common mechanism. This issue is covered in more detail in the Security Considerations section of this document.

3.1.3 CAPABILITY

Arguments: none

Data: Server capability information as described below

Result: 2.0 - authenticate completed, now in authenticated state

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. The response may differ depending on the current state of the connection. The responses may also differ depending upon the authenticated user.

Sender implementations SHOULD NOT require any capability name beyond those defined in this specification, and MUST tolerate any unknown capability names. This command may return different results in the Connected states versus the Authenticated state. It may also return different results depending on the authenticated calendar user.

The format of the capabilities response is a series of lines with the form <name>[=<value>]. Each name-value pair is delimited by a <CRLF> character sequence. Each line, including the terminating <CRLF> MUST be 1024 characters or less. The sequence <CRLF>.<CRLF> followed by a reply code terminates the response.

The table below summarizes the information available in response to a CAPABILITY command.

Capability	Occurs	Description
iRIPrev1	1	Revision of iRIP, must be "iRIPrev1"
AUTH	0+	Authentication mechanism(s) supported
MAXICALOBJECTSIZE	0 or 1	An integer value that specifies the largest ICAL object (byte count) the server will accept. Objects larger

than this will be rejected.

MAXDATE	0 or 1	The datetime value beyond which the server cannot accept.
MINDATE	0 or 1	The datetime value prior to which the server cannot accept.

Examples:

When executed from the Connected state the following occur:

```
S: CAPABILITY
R: iRIPrev1
R: AUTH=KERBEROS_V4
R: AUTH=PLAIN
R: .
R: 2.0
```

When executed from the Authenticated state the following occur:


```
S: CAPABILITY
R: CAPABILITY iRIPrev1
R: AUTH=KERBEROS_V4
R: AUTH=PLAIN
R: MAXICALOBJECTSIZE=50000
R: .
R: 2.0
```

3.1.4 CONTINUE

```
Arguments:  [latencyTime]
Data:       noneResult:     results from the command in progress
            2.0.2  reply pending.
            9.1    unexpected command
```

The CONTINUE command is issued by the sender to allow an SENDATA request to continue being processed. When the latency time is specified on the SENDATA command, the Receiver must issue a reply to the Sender within the specified time. The reply could 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 command in progress.

The CONTINUE has the following form:

```
     CONTINUE [latencyTime]
```

If the optional latencyTime is present, it is a positive integer that specifies the maximum number of seconds the client will wait for the next response. If it is omitted, the receiver waits an indefinite period of time for the response.

In this example, the sender requests a response from the server every **10 seconds.**

```
...
S: RECIPIENT irip://A.example.com/sman
R: 2.0
S: SENDATA 10
R: 2.0.1
S: Content-Type:text/calendar; method=REQUEST; charset=US-ASCII
S: Content-Transfer-Encoding: 7bit
S:
S: BEGIN:VCALENDAR
# etc...
S: END:VCALENDAR
S: .

#  after 10 seconds...
```



```
R: .
R: 2.0.2  Reply Pending
S: CONTINUE 10

#  less than 10 seconds elapse...

R: 2.0.11 irip://A.example.com/sman
R: Content-Type:text/calendar; method=REPLY; charset=US-ASCII
R: Content-Transfer-Encoding: 7bit
R:
R: BEGIN:VCALENDAR
# etc...
R: END:VCALENDAR
R: .
```

3.1.5 DEQUEUE

Arguments: caluid [latencyTime]

```
Data:       a single queued itip message on success.

Result:     2.0 <caluid>   success
            2.0.2 <caluid> reply pending.
            2.0.8 <caluid> no more messages
            3.8 <caluid>   no authority to perform this operation
            9.1            unexpected command
```

The DEQUEUE command requests the Receiver to send a single queued message to the Sender. This differs from using the SWITCH command in several ways:

- the SWITCH command results in the Connected state after the Sender and Receiver roles are reversed. This means that both the Sender and Receiver must be prepared to handle the AUTHENTICATE command. Using the DEQUEUE command, queued commands can be collected by the original Sender without it having to handle the AUTHENTICATE command.
- Only one message is transferred per DEQUEUE command.
- The single and implicit receiver of a DEQUEUE message is the currently authenticated Sender.

If the Receiver has a queued message for the CALID and the authenticated user is allowed to access the queue, it will be sent as the reply to the DEQUEUE message. The message is followed by a (required) <CRLF>.<CRLF> and a (required) response code.

Example:

```
S: DEQUEUE irip://cal.example.com/sman
R: Content-Type:text/calendar; method=REQUEST; charset=US-ASCII
R: Content-Transfer-Encoding: 7bit
```



```
R:
R: BEGIN:VCALENDAR
#  etc...
R: END:VCALENDAR
R: .
R: 2.0 irip://cal.example.com/sman
S: DEQUEUE irip://cal.example.com/sman
R: .
R: 2.0.8 irip://cal.example.com/sman
```

3.1.6 DISCONNECT

Arguments: none

Data: none

Result: 2.0 - success

The DISCONNECT command signals the end of communication between the Sender and Receiver. It SHOULD be issued only from the Authenticated or Connected states. However, receiver implementations MUST be prepared to handle a DISCONNECT at any point in this state diagram.

Example:

S: DISCONNECT
R: 2.0

3.1.7 RECIPIENT

Arguments: caluid [latencyTime]

Data: none

Result: 2.0 <caluid>   - Found the calendar

            2.0.4 <caluid> - caluid is not on this irip server but an attempt will be made to deliver the request or reply to the Calendar anyway. A trust relationship exists with IRIP server for caluid.

            2.0.5 <caluid> - just like 2.0.4 except that the message MUST be queued. That is, it will not be possible for this request to be processed in real-time.

            2.0.6 <caluid> - The specified <caluid> is not here but an attempt will be made to deliver the request or reply to <caluid> anyway.

There is not a trust relationship between the IRIP server and the IRIP server for the target calendar.

            3.8 <caluid>   - the authenticated iRIP session does not have authority to perform [iTIP] activities on <caluid>.

```
5.3 <caluid>   - IRIP services for the specified calid
                 are not supported.

9.1            - Unexpected command

10.1 <caluid> <newcaluid>
               - <caluid> is not supported by this iRIP
                 service but can be found <newcaluid>.
                 Note that <newcaluid> need not be of the
                 IRIP scheme.
```

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

A reply code will be returned for each calendar address supplied. A response code for each calendar address will also be returned after the SENDATA command completes.

A reply code of 2.0 indicates that the calendar address is available for [ITIP] messages. If the receiver does not accept [ITIP] messages for the specified calendar address, it responds with [ITIP] reply code **5.3 to indicate that the calendar address is unknown. If the receiver** has a referral calendar address it responds with reply code 10.1 and supplies the new calendar address. In either case, the iRIP server does not deliver the [ITIP] message when the reply code is 5.3 or 10.1.

After the ITIP message has been sent, a reply code will be returned for each of the recipients.

3.1.7.1 Fanout Issues On RECIPIENT

A receiver may be implemented such that it will fanout requests to other iRIP servers. That is, a sender connects to iRIP receiver at A.example.com and specifies a RECIPIENT calendar address on B.example1.com. If the iRIP server at A.example.com handles getting the request to the receiver at B.example1.com it supports fanout.

```
              iRIP                          iRIP
    Sender ----------- [A.example.com] ------------- [B.example.com]
```

An iRIP server implementation can implement fanout in different ways.

One way involves verifying remote recipient calendars in real-time.

Another way saves up all remote recipient calendars and simply attempts to access them later. The advantage of verifying remote calendars in real-time is that the sender is notified immediately, via the reply code, whether or not the recipient calendar exists and is accessible. For example, suppose that the iRIP server on A.example.com just received the following command from Sender:

RECIPIENT irip://b.example.com/sam

If A.example.com immediately contacts B.example.com and issues a RECIPIENT irip://b.example.com/sman and returns the reply back to Sender, the sender will have an authoritative reply code (2.0 - success, 3.7 - invalid calendar, or 10.1 - referral). On the other hand, if A.example.com simply collects all the remote calendar addresses and attempt to access them later in the transaction the reply will be 2.0.4 (will attempt). The disadvantage of this approach is that the sender does not know the status of the target calendar during the RECIPIENT negotiation.

3.1.8 SENDATA

Arguments: [latencyTime]

Data: MIME encapsulated iCalendar object

Result: 2.0.1 - Begin sending the MIME encapsulated iCalendar Object
9.1 - Unexpected command

After sending the iCalendar object a result is returned for each recipient. These results can be the following:

2.0 <caluid> - Success. [ITIP] message delivered.
2.0.2 <caluid> - A timeout has occurred.
2.0.3 <caluid> - In response to the client issuing an ABORT
2.0.7 <caluid> - The message has been queued for delivery.
2.0.11 <caluid> - Success. [ITIP] message delivered and a response follows.

8.0 A failure has occurred in the receiver that prevents the operation from succeeding.

8.1 Sent when a session cannot be established because the iRIP Receiver is too busy.

8.2 Used to signal that an ICAL object has exceeded the server's size limit.

9.0 An unrecongnized command (METHOD) was received.

9.1 A command was issued in a manner inconsistent with the state diagram. For example, issuing the SENDATA command without having specified a RECIPIENT will cause this error.

10.2 The server is shutting down.

10.4 The operation has not be performed because it would cause the resources (memory, disk, CPU, etc) to exceed the allocated quota

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

```
S: SENDATA [latencyTime]
R: 2.0.1
S: <MIME encapsulated sender ITIP Message>
S: .
R: <reply code> <caluid>

#  if the reply code above is 2.0.11 the following will also be sent:

R: <MIME encapsulated reply ITIP Message.>
R: .
```

The optional latencyTime value specifies the maximum number of seconds the sender will wait for a reply. If it is not present, the client places no time limit on the server for a reply. A reply code of 2.0.1 indicates that the [ITIP] message data can be sent. The data must be broken into lines that are 1024 characters (including the ending <CRLF> or less. When the entire message has been sent, the sender terminates sending data with the special sequence <CRLF>.<CRLF>. The receiver reply MAY contain an [ITIP] message. The reply MUST contain the special sequence <CRLF>.<CRLF> followed by a reply code for each RECIPIENT.

The command sequence for an iRIP server that does not include an [ITIP] message in the reply might appear as follows:

```
S: RECIPIENT irip://cal.example.com/johndoe
R: 2.0
S: RECIPIENT irip://cal.othersystem.com/xyz
R: 2.0.5
S: SENDATA
```

```
R: 2.0.1
# lots of data
S: .
R: .
R: 2.0 irip://cal.example.com/johndoe
```



```
R: 2.0.7 irip://cal.othersystem.com/xyz
```

If the reply code is 2.0.11, an [ITIP] message reply will follow. This message will be terminated by the <CRLF>.<CRLF> sequence. iRIP servers are not required to send [ITIP] messages in the reply to [ITIP] requests delivered via the SENDATA command. However, the protocol allows for high performance servers to do so. iRIP senders MUST accept the [ITIP] message if the receiver includes it the reply.

3.1.9 SWITCH

Arguments: none

Data: none

Result:	2.0	Sender and receiver have switched roles. The connection is switched to the Connected State.
	3.14	Unsupported command. That is, the receiver refuses to switch roles.

The SWITCH command is used to allow the Sender and Receiver to change roles. After a switch command is executed and the new Sender authenticates, all queued commands that the new Sender has queued for the new Receiver will be delivered.

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:

- send an OK reply and take on the role of Sender
- 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. The IRIP connection returns to the Connected state. Program-A is then in its initial state and sends a service ready response code of 2.0.

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 (connected) as if it had just connected to Program-A, and expects to receive a response code of 2.0.

3.2 Fanout and Queued Transactions

An iRIP server must be able to fanout requests targeted at other iRIP servers. An iRIP server may queue information targeted at other iRIP servers. There are several reasons for queing requests. One reason is that firewall issues may prevent one server from contacting another.

iRIP servers can establish trust relationships between each other. A trusted relationship means:

- one server must authenticate with the other
- authenticated calendars on one server are trusted and treated as authenticated on the other.

The trusted relationship need not be bi-directional. That is, the fact that iRIP server A trusts iRIP server B does not necessarily mean that B trusts A.

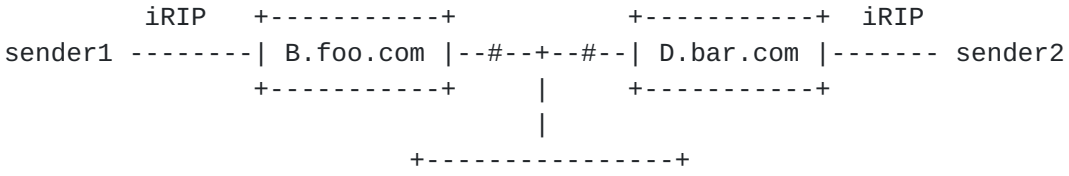
A trusted relationship between two iRIP servers means that one server can queue transactions for the other server and deliver them some time later. If iRIP server B trusts A, then A can queue requests for B. If A does not trust B then B cannot accumulate requests for A. [Editors Note: do we really want to impose this restriction?]

Certain requests may need to be delivered and replied to in real-time. In fact, a requester may wish to cancel the request if the reply cannot be delivered in real-time. In iRIP the reply code to the RECIPIENT command indicates whether or not a reply will be made in real-time (barring connection and hardware failures). This allows the sender to abort the request if necessary.

3.3 Bi-Directional Queue Operation

It is possible that firewall configurations may not allow a connection between two iRIP servers in either direction. That is, in the diagram below, suppose there are two users, sender1 and sender2, who wish to exchange [ITIP] messages. Their calendar addresses are

irip://B.foo.com/sender1 and irip://D.bar.com/sender2. The firewall in front of B.foo.com prevents incoming external connections on the iRIP server port. However, it allows outbound external connections on the iRIP server port to happen. Similarly, the firewall in front of D.bar.com also prevents inbound connections on the iRIP server port, but allows outbound connections. To allow sender1 and sender2 to exchange iRIP messages an intermediate iRIP server, C.foobar.com, is used to queue messages for both of their calendars. A trust relationship between the intermediate iRIP server and the endpoint servers (B.foo.com and D.bar.com) is desirable but not required. [Editors note: is desired but not required OK with everyone?]



3.4 Reply 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:

REPLY CODE	DESCRIPTION	MEANING
2.0	**STATOK**	Operation was successfully performed.
2.0.1	**START-SENDATA**	Start ICAL input; end with <CRLF>.<CRLF>
2.0.11	**OK-DATAFOLLOWS**	The request was processed successfully. Reply data follows on the next line and terminates 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.

2.0.3	**ABORTED**	In response to the client issuing an ABORT command, this reply code indicates that any command currently underway was successsfully aborted.
2.0.4	**WILL-ATTEMPT**	The specified Calendar is not here but an attempt will be made to deliver the request or reply to the Calendar anyway. There is a trust relationship between this iRIP server and the iRIP server for the target calendar.
2.0.5	**TRUSTED-WILL-QUEUE**	The specified Calendar cannot be contacted directly and a trust relationship exists between this server and the server on which the Calendar exists. The request or reply will be queued and delivered to the target calendar when its iRIP server contacts this server and issues the SWITCH command.

2.0.6	**WILL-ATTEMPT**	The specified Calendar is not here but an attempt will be made to deliver the request or reply to the Calendar anyway. There is not a trust relationship between the iRIP server and the iRIP server for the target calendar.
2.0.7	**QUEUED**	The message has been queued for delivery.
2.0.8	**QUEUE-EMPTY**	There are no more queued messages.
2.2	**NO COMMAND IN PROGRESS**	An ABORT or CONTINUE was received when no command was in progress
6.1	**AUTHENTICATE FAILURE**	Unsupported authentication mechanism, credentials rejected
6.2	**AUTHENTICATION ABORTED**	Sender aborted authentication, authentication exchange cancelled
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 session cannot be established because the iRIP Receiver is too busy.
8.2	**ICAL OBJECT TOO BIG**	Used to signal that an ICAL object has exceeded the server's size limit.
8.3	**DATE TOO LARGE**	A DATETIME value was too far in the future to be represented on this Calendar.
8.4	**DATE TOO SMALL**	A DATETIME value was too far in the past to be represented on this Calendar.
9.0	**INVALID iRIP COMMAND**	An unrecongnized command was received.
9.1	**UNEXPECTED COMMAND**	A command was issued in a manner inconsistent with the state diagram. For example, issuing the SENDATA command without having specified any RECIPIENTs will cause this error.
10.1	**REFERRAL**	Accompanied by an alternate address. The RECIPIENT specified should be contacted at the given alternate

		address. The referral address MUST follow the reply code.
10.2	**SERVER SHUT DOWN**	The server is shutting down.
10.3	**SERVER STOPPING FLOOD 2**	
10.4	**EXCEEDED QUOTAS**	The operation has not be performed because it would cause the resources (memory, disk, CPU, etc) to exceed the allocated quota
10.5	**QUEUED TOO LONG**	The ITIP message has been queued too long. Delivery has been aborted.

4 Implementation Considerations

It is strongly recommended that when an iRIP implementation encounters an error requiring the communication channel between the Sender and Receiver to be dropped that the DISCONNECT command be issued rather than simply breaking the communication channel.

5 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 transactions, it is recommended that implementations use the highest degree of authentication and transmission security possible.

5.1 Security Threats and Recommendations

In addition to the security risks detailed in [ITIP], the following sections discuss security risks in using iRIP as the transport binding.

5.1.1 Authentication Hacking

Once authenticated, senders can re-authenticate from the Authenticated state. It is possible that, once authenticated, a sender could take advantage of this capability and repeatedly attempt to guess at calendar user credentials. It is recommended that implementations disconnect after a failed authentication attempt from the Authenticated state.

5.1.2 Spoofing

The [ITIP] paradigm allows any modifications to data by its Organizer, which maps to the [SASL] authorization identity. This means that authorizing with appropriate identities over iRIP will allow read and write access to any item in the Receiver's database. There are several ways to limit this security risk.

The choice of accepted authentication mechanisms can reduce the security risk. The ANONYMOUS mechanism allows a greater level of interoperability, in that any Sender can connect anonymously, but

greatly increases the security risk for the same reason.

The method in which authorizations are accepted can also be modified to improve security. Some hosts may be trusted to authorize as any caladdress, while others may be only be trusted to authorize as users in their domain.

[ITIP] data is transmitted in MIME containers, which provide a facility for digitally signing their data. A sender may use this scheme in order to provide a final security fallback.

Finally, some implementations may decide to queue incoming iRIP commands for approval by the owner of the calendar, although this is certainly the least reliable of these security mechanisms.

5.1.3 Eavesdropping

The use of SASL in iRIP allows the negotiation of an encrypted security layer, which greatly reduces the chances that a connection will be subject to eavesdropping.

However, if another iRIP server is being used to relay iRIP data this relay server is privy to whatever information is being transmitted. For this reason it may be desirable to use MIME's encryption facility to protect the data.

5.1.4 Connection Flooding

Servers should be configurable to timeout unused connections.

5.2 Security Interoperability Issues

* minimum SASL mechanism
 [ed note: tbd Paul Hill to supply]
* add a failure case under 6.3

6 Examples

6.1 Unauthenticated Freebusy Request

This examples shows an anonymous request for the freebusy time of irip://cal.example.com/sman. Note that once xyz is authenticated on the IRIP server either the fully qualified IRIP CALID or the relative CALID can be used to reference a Calendar. That is,

"irip://cal.example.com/xyz" and "xyz" refer to the same calendar and
can be used interchangeably.

```
R: <listen on TCP port 5228>
S: <establish a TCP connection to cal.example.com port 5228>
R: 2.0
S: AUTHENTICATE ANONYMOUS
R: 2.0
S: RECIPIENT:irip://b.foo.bar/sman
R: 2.0
S: SENDATA
R: 2.0.1
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: irip://b.foo.bar/xyz
S: ATTENDEE: irip://b.foo.bar/sman
S: DTSTAMP:19971113T190000Z
S: DTSTART:19971115T160000Z
S: DTEND:19971116T040000Z
S: UID:www.example.com-873970198738777@host.com
S: END:VFREEBUSY
S: END:VCALENDAR
S: .

# server looks up the freebusy time and builds a reply>

R: 2.0.11 irip://cal.example.com/sman
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: ORGANIZER:irip://cal.example.com/xyz
R: ATTENDEE:irip://cal.example.com/sman
R: DTSTAMP:19971113T190005Z
R: DTSTART:19971115T160000Z
R: DTEND:19971116T040000Z
R: UID:www.example.com-873970198738777@host.com
```

```
R: FREEBUSY:19971115T230000Z/PT1H,19971115T210000Z/PT30M
R: END:VFREEBUSY
R: END:VCALENDAR
R: .
S: DISCONNECT
R: 2.0
R: <disconnect>
S: <disconnect>
```

6.2 Busy Time Request

In this example, the sender sends a Freebusy request to the iRIP server at B.foo.com for several calendars. Some of the calendars are on other calendar stores. The sender needs the information immediately and will abort any attempt to queue requests.

```
R: <listen on TCP port 5228>
S: <establish a TCP connection to B.foo.com port 5228>
R: 2.0
S: AUTHENTICATE KERBEROS_V4 93407205

#  more authentication information

R: 2.0
S: RECIPIENT:irip://B.foo.com/bill
R: 2.0
S: RECIPIENT:irip://C.foobar.com/cathy
R: 2.0.4
S: RECIPIENT:irip://D.bar.com/david
R: 2.0.5
S: RECIPIENT:irip://E.barfoo.com/eddie
R: 2.0.6

#  the sender does not want this ITIP message to be queued request.
#  So, the current operation will be canceled. The operation will be
#  tried again with attendees that can be serviced in real-time.

S: ABORT
R: 2.0.3
S: RECIPIENT:irip://B.foo.com/bill
R: 2.0
S: RECIPIENT:irip://C.foobar.com/cathy
R: 2.0.4
S: RECIPIENT:irip://E.barfoo.com/eddie
R: 2.0.6
S: SENDATA
R: 2.0.1
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:irip://B.foo.com/bill
S: ATTENDEE:irip://B.foo.com/bill
S: ATTENDEE:irip://C.foobar.com/cathy
S: ATTENDEE:irip://D.bar.com/david
S: ATTENDEE:irip://E.barfoo.com/eddie
S: DTSTAMP:19971113T190000Z
S: DTSTART:19971115T160000Z
S: DTEND:19971116T040000Z
S: UID:www.example.com-873970198738777@host.com
S: END:VFREEBUSY
S: END:VCALENDAR
S: .

# server looks up the freebusy time for irip://B.foo.com/bill,
# requests and receives the freebusy time for
# irip://C.foobar.com/cathy and irip://E.barfoo.com/eddie. Then it
# builds a reply

R: 2.0.11 irip://B.foo.com/bill
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
S: ORGANIZER:irip://B.foo.com/bill
R: ATTENDEE:irip://B.foo.com/bill
R: DTSTAMP:19971113T190005Z
R: DTSTART:19971115T160000Z
R: DTEND:19971116T040000Z
R: UID:www.example.com-873970198738777@host.com
R: FREEBUSY:19971115T200000Z/PT1H,19971116T030000Z/PT30M
R: END:VFREEBUSY
R: END:VCALENDAR
R: .
R: 2.0.11 irip://C.foobar.com/cathy
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
S: ORGANIZER:irip://B.foo.com/bill
R: ATTENDEE:irip://C.foobar.com/cathy
R: DTSTAMP:19971113T190005Z
```



```
R: DTSTART:19971115T160000Z
R: DTEND:19971116T040000Z
R: UID:www.example.com-873970198738777@host.com
R: FREEBUSY:19971115T230000Z/PT1H,19971116T020000Z/PT30M
R: END:VFREEBUSY
R: END:VCALENDAR
R: .
R: 2.0.11 irip://E.barfoo.com/eddie
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
S: ORGANIZER:irip://B.foo.com/bill
R: ATTENDEE:irip://E.barfoo.com/eddie
R: DTSTAMP:19971113T190005Z
R: DTSTART:19971115T160000Z
R: DTEND:19971116T040000Z
R: UID:www.example.com-873970198738777@host.com
R: FREEBUSY:19971115T230000Z/PT1H,19971116T020000Z/PT30M
R: END:VFREEBUSY
R: END:VCALENDAR
R: .
S: DISCONNECT
R: 2.0
R: <disconnect>
S: <disconnect>
```

6.3 Using Switch

This session demonstrates how a poll can be accomplished using the

SWITCH command. In this case, the receiver (A.example.com) becomes
the sender after issuing the switch command.

```
# receiver is currently A.example.com, the sender is B.xyz.com
R: <listen on TCP port 5228>
S: <establish a connection to TCP port 5228>
R: 2.0
S: AUTHENTICATE KERBEROS_V4
#  more authentication...
R: 2.0
S: SWITCH
R: 2.0

#  The connection state has returned to the Connected state.
A.example.com
#  must now Authenticate with B.xyz.com
```



```
S: 2.0
R: AUTHENTICATE KERBEROS_V4
#  more authentication...
R: 2.0

#  Now that the switch has occurred, A.example.com is the sender and
#  B.xyz.com is the receiver.  At this point, A.example.com sends all
#  queued commands for recipients on B.xyz.com

S: RECIPIENT:irip://B.xyz.com/sman
R: 2.0
S: SENDATA
R: 2.0.1
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:irip://A.example.com/billybob
S: ATTENDEE:irip://B.xyz.com/sman
S: DTSTAMP:19981113T190000Z
S: DTSTART:19981115T160000Z
S: DTEND:19981116T160000Z
S: UID:123456abcdef.1234.2314
S: END:VFREEBUSY
```

```
S: END:VCALENDAR
S: .

# server looks up the freebusy time and builds a reply

R: 2.0.11 irip://B.xyz.com/sman
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: ORGANIZER:irip://cal.example.com/xyz
R: ATTENDEE:irip://cal.example.com/sman
R: DTSTAMP:19981113T190005Z
R: DTSTART:19981115T160000Z
R: DTEND:19981116T160000Z
R: UID:123456abcdef.1234.2314
R: FREEBUSY:19981115T230000Z/PT1H,19981115T210000Z/PT30M
R: END:VFREEBUSY
R: END:VCALENDAR
R: .
```

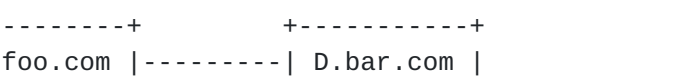


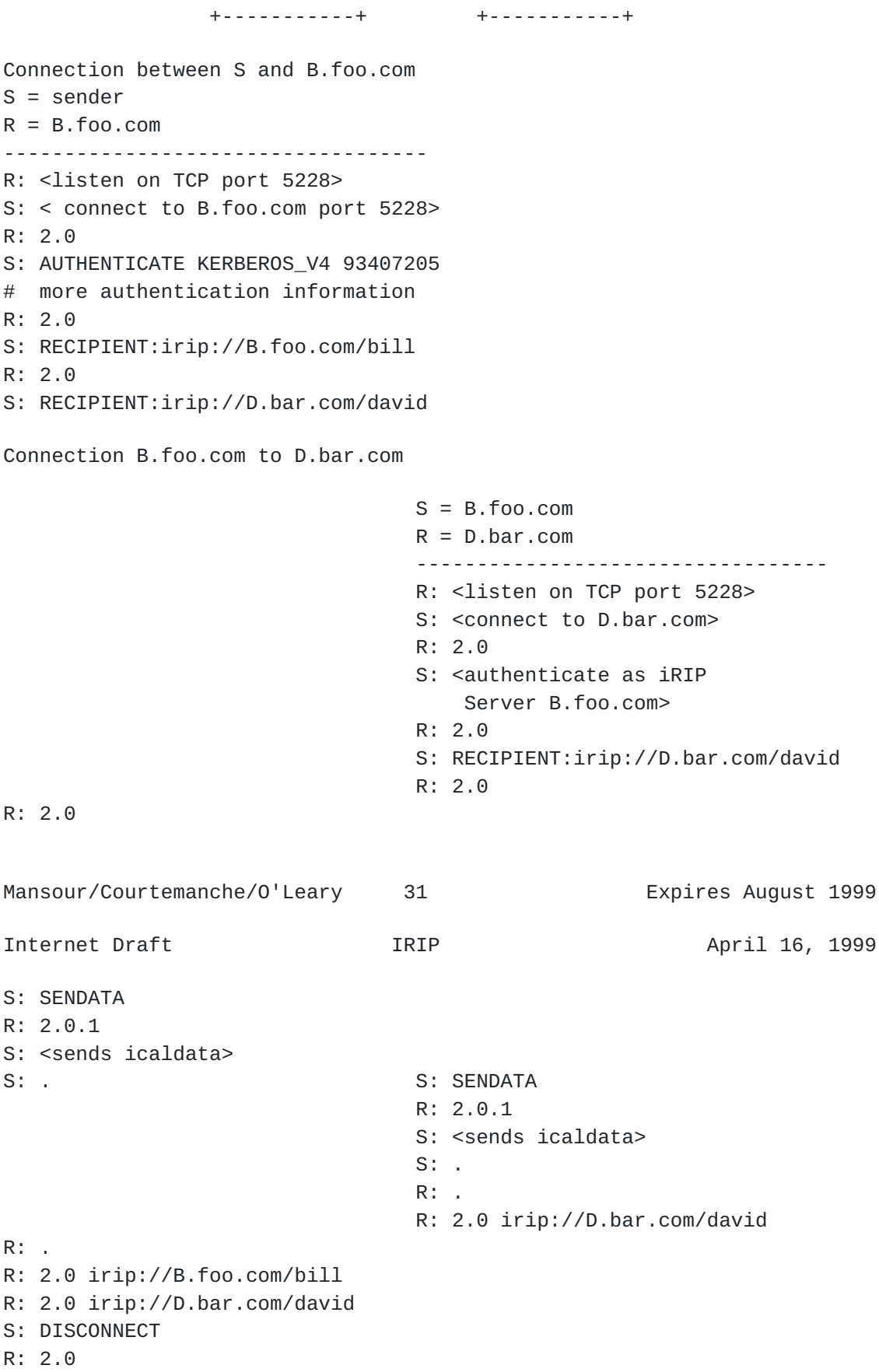
```
#  A.example.com has no more queued ITIP messages for B.xyz.com.
#  So, it disconnects...

S: DISCONNECT
R: 2.0
R: <disconnect>
S: <disconnect>
```

6.4 Fanout Requests

6.4.1 Successful Fanout Request

In the diagram below, sender has authenticated to the iRIP server B.foo.com and is attempting to deliver a request to calendars on B.foo.com and C.foobar.com. The iRIP server on B.foo.com supports fanout. It verifies remote calendars during the RECIPIENT negotiation with sender.

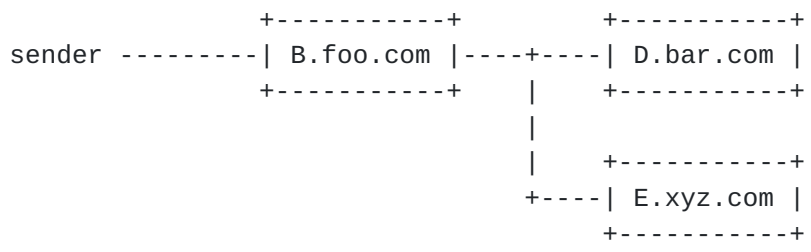



+-----------+ +-----------+
Connection between S and B.foo.com
S = sender
R = B.foo.com
R: <listen on TCP port 5228>
S: < connect to B.foo.com port 5228>
R: 2.0
S: AUTHENTICATE KERBEROS_V4 93407205
more authentication information
R: 2.0
S: RECIPIENT:irip://B.foo.com/bill
R: 2.0
S: RECIPIENT:irip://D.bar.com/david
Connection B.foo.com to D.bar.com
S = B.foo.com
R = D.bar.com
R: <listen on TCP port 5228>
S: <connect to D.bar.com>
R: 2.0
S: <authenticate as iRIP
Server B.foo.com>
R: 2.0
S: RECIPIENT:irip://D.bar.com/david
R: 2.0
R: 2.0

S: SENDATA
R: 2.0.1
S: <sends icaldata>
S: .
S: SENDATA
R: 2.0.1
S: <sends icaldata>
S: .
R: .
R: 2.0 irip://D.bar.com/david
R: .
R: 2.0 irip://B.foo.com/bill
R: 2.0 irip://D.bar.com/david
S: DISCONNECT
R: 2.0

6.4.2 Referral On Fanout

This example is just like the one above except that in this case the remote calendar no longer exists and a referral is returned. The sender cancels the transaction in the RECIPIENT phase (using ABORT) and starts a new transaction that uses the referral address.




```
Connection between S and B.foo.com
S = sender
R = B.foo.com
====================================
S: < connect to B.foo.com port 5228>
R: 2.0
S: AUTHENTICATE KERBEROS_V4 93407205
S: <more authentication information>
R: 2.0
S: RECIPIENT:irip://B.foo.com/bill
R: 2.0
S: RECIPIENT:irip://D.bar.com/david
```

```
                                  Connection B.foo.com to D.bar.com

                                  S = B.foo.com
                                  R = D.bar.com
                                  ====================================

                                  R: <listen on TCP port 5228>
```



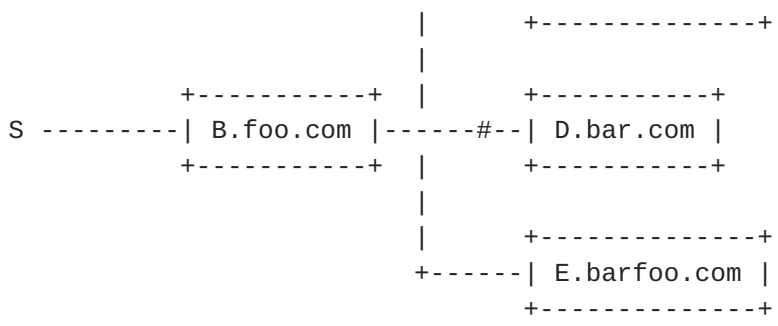
```
                                  S: <connect to D.bar.com port 5228>
                                  R: 2.0
                                  S: <authenticates as irip server
                                      B.foo.com>
                                  R: 2.0
                                  S: RECIPIENT:irip://D.bar.com/david
                                  R: 10.1 irip://E.xyz.com/david99
R: 10.1 irip://E.xyz.com/david99
S: ABORT
```

```
R: 2.0.3
S: RECIPIENT:irip://B.foo.com/bill
R: 2.0
S: RECIPIENT irip://E.xyz.com/david99

                                Connection B.foo.com to E.xyz.com
                                S = B.foo.com
                                R = E.xyz.com
                                ==================================
                                S: <connect to D.bar.com port 5228>
                                R: 2.0
                                S: <authenticates as irip server
                                    B.foo.com>
                                R: 2.0
                                S: RECIPIENT irip://E.xyz.com/david99
                                R: 2.0
R: 2.0
S: SENDATA
R: 2.0.1
S: <sends icaldata>
S: .                            S: SENDATA
                                R: 2.0.1
                                S: <sends icaldata>
                                S: .
                                R: .
                                R: 2.0 irip://E.xyz.com/david99
R: .
R: 2.0 irip://B.foo.com/bill
R: 2.0 irip://E.xyz.com/david99
S: DISCONNECT
R: 2.0
```

6.5 Queued Requests

In the diagram below, sender S has authenticated to the iRIP server B.foo.com. C.foobar.com, D.bar.com, and E.barfoo.com all have iRIP servers. The iRIP server on B.foo.com has a trusted relationship with iRIP servers on both C.foobar.com and D.bar.com. A firewall is in place that prohibits iRIP server B.foo.com from initiating a connection to the iRIP server on D.bar.com. However, iRIP server D.bar.com can connect to iRIP server B.foo.com.


```
                              +--------------+
                        +-----| C.foobar.com |
```



6.5.1 Meeting Invitation

In this example, S sends an event request to the iRIP server on B for calendars on B, C, D, and E. Note that C's address moved from foo.com to foobar.com and is reported to the sender during the RECIPIENT negotiation.

```
R: <listen on TCP port 5228>
S: <establish a TCP connection to b.foo.com port 5228>
R: 2.0
S: AUTHENTICATE KERBEROS_V4 93407205
S: <more authentication information>
R: 2.0
S: RECIPIENT irip://B.foo.com/bill
R: 2.0
S: RECIPIENT irip://C.foobar.com/cathy
R: 10.1 irip://C.foobar.com/cathy
S: RECIPIENT irip://C.foobar.com/cathy
R: 2.0.4
S: RECIPIENT irip://D.bar.com/david
R: 2.0.5
S: RECIPIENT irip://E.barfoo.com/eddie
R: 2.0.6
S: SENDATA 16
R: 2.0.1
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:VEVENT
S: ORGANIZER:irip://B.foo.com/bill
S: ATTENDEE;ROLE=CHAIR;PARTSTAT=ACCEPTED:irip://B.foo.com/bill
S: ATTENDEE;RSVP=TRUE;TYPE=INDIVIDUAL:irip://C.foobar.com/cathy
S: ATTENDEE;RSVP=TRUE;TYPE=INDIVIDUAL:irip://D.bar.com/david
S: ATTENDEE;RSVP=TRUE;TYPE=INDIVIDUAL:irip://E.barfoo.com/eddie
S: DTSTAMP:19981011T190000Z
S: DTSTART:19981101T200000Z
```

```
S: DTEND:19981101T2100000Z
S: SUMMARY:Conference
S: UID:calsrv.example.com-873970198738777@example.com
S: SEQUENCE:0
S: STATUS:CONFIRMED
S: END:VEVENT
S: END:VCALENDAR
S: .
R: .
R: 2.0 irip://B.foo.com/bill
R: 2.0 irip://C.foobar.com/cathy
R: 2.0.7 irip://D.bar.com/david
R: 2.0 irip://E.barfoo.com/eddie
S: DISCONNECT
R: 2.0
R: <disconnect>
S: <disconnect>
```

The invitation is written to the calendar B.foo.com/bill. iRIP server B.foo.com authenticates to C.foobar.com and sends the event request, which is successfully written to C.foobar.com/cathy. The iRIP server on B.foo.com cannot contact D.bar.com, but a trust relationship exists between them and the request is queued for delivery. This request will be delivered the next time the iRIP server on D.bar.com connects to the iRIP server on B.foo.com and issues a SWITCH command. The iRIP server on B.foo.com connects to the iRIP server on E.barfoo.com and authenticates as anonymous since it has no trust relationship with E.barfoo.com. If the anonymous authentication is successful, the event request is delivered to E.barfoo.com/eddie.

7 Acknowledgments

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

Frank Dawson, Bruce Kahn, Doug Royer, Mugino Saeki

8 Bibliography

[ICAL] F. Dawson, D. Stenerson, "Internet Calendaring and Scheduling Core Object Specification - iCalendar", RFC-2445, November 1998, http://www.imc.org/rfc2445.

[ITIP] S. Silverberg, S. Mansour, F. Dawson, R. Hopson, "iCalendar Transport-Independent Interoperability Protocol (iTIP) : Scheduling Events, Busy Time, To-dos and Journal Entries", RFC-2446, November 1998, http://www.imc.org/rfc2446.

[IMIP] F. Dawson, S. Mansour, S. Silverberg, "iCalendar Message-based Interoperability Protocol (iMIP), ", RFC-2447, November 1998, http://www.imc.org/rfc2447.

[ID-UTF8] 3"UTF-8, a transformation format of ISO 10646. F. Yergeau.",

RFC 2299, January 1998

[RFC-822] Crocker, D., "Standard for the Format of ARPA Internet Text Messages", STD 11, RFC 822, August 1982.

[RFC-1847]. J. Galvin, S. Murphy, S. Crocker & N. Freed, "Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted", RFC 1847, October 1995.

[RFC-2112] Levinson, E., "The MIME Multipart/Related Content-type," RFC 2112, March 1997.

[RFC-2015] M. Elkins, "MIME Security with Pretty Good Privacy (PGP)," RFC 2015, October 1996.

[RFC-2045] Freed, N., Borenstein, N., " Multipurpose Internet Mail Extensions (MIME) - Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

[RFC-2046] Freed, N., Borenstein, N., " Multipurpose Internet Mail Extensions (MIME) - Part Two: Media Types", RFC 2046, November 1996.

[RFC-2047] Moore, K., "Multipurpose Internet Mail Extensions (MIME) - Part Three: Message Header Extensions for Non-ASCII Text", RFC 2047, November 1996.

[RFC-2048] Freed, N., J. Klensin, J. Postel, "Multipurpose Internet Mail Extensions (MIME) - Part Four: Registration Procedures", RFC 2048, January 1997.

[RFC-2222] J. Meyers, Simple Authentication and Security Layer (SASL)", RFC 2222, October 1997.

[RFC2396] Berners-Lee, Fielding, Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998.

[RFC-2445] Dawson, Stenerson, "Internet Calendaring and Scheduling Core Object Specification (iCalendar)", RFC 2445, November 1998

[RFC-2446] Silverberg, Mansour, Dawson, Hopson, "iCalendar Transport-Independent Interoperability Protocol (iTIP)", RFC 2446, November 1998

[RFC-2447] Dawson, Mansour, Silverberg, "iCalendar Message-Based Interoperability Protocol (iMIP)", RFC 2445, November 1998

9 Open Issues

Registration of the [SASL] profile for iRIP with the IANA.
Port Number registration

10 Author's Address

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

```
BEGIN:VCARD
VERSION:2.1
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: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:Pete O'Leary
ORG:Amplitude
ADR;WORK;POSTAL;PARCEL:;;
TEL;WORK;MSG:+1-415-659-3511
TEL;WORK;FAX:+1-415-659-0006
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 co-chair of that working group is:

BEGIN:VCARD
FN:Pat Egen
ORG:Egen Consulting
ADR;WORK;POSTAL;PARCEL:;;803 Creek Overlook;Chattanooga;TN;37415
TEL;WORK;MSG:+1-423.875.2652
TEL;WORK;FAX:+1-423.875.2017
EMAIL;INTERNET:pregen@engenconsulting.com
END:VCARD

11 Full Copyright Statement