INTERNET-DRAFT
Expires: December 16, 1998

Dalen Abraham
Microsoft Corporation

June 1998

Extensions to the Internet Relay Chat Protocol (IRCX)
draft-pfenning-irc-extensions-04.txt

1. 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 and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

To view the entire list of current Internet-Drafts, please check the "1id-abstracts.txt" listing contained in the Internet-Drafts Shadow Directories on ftp.is.co.za (Africa), ftp.nordu.net (Northern Europe), ftp.nis.garr.it (Southern Europe), munnari.oz.au (Pacific Rim), ftp.ietf.org (US East Coast), or ftp.isi.edu (US West Coast).

2. Abstract

This document describes IRCX, a set of extensions to the Internet Relay Chat (IRC) protocol defined in RFC 1459[1]. Only client-server interactions are defined. The added functionality includes user authentication for multiple security providers, support for UNICODE[2] characters, multilayer security, and a unified property mechanism. Chat server implementations can support channel or server services with full control over all messages and events. These services communicate with the core server over an extended IRC connection.

All changes to the IRC protocol are designed such that existing clients will continue to work against servers implementing the extensions. Support for the extended protocol can be queried by clients to take advantage of the added functionality or to conform to the basic protocol as defined in RFC1459.

2.1. Contents

3. Modified UTF8 Encoding of UNICODE characters

Allowing UNICODE characters for all user visible strings introduces a set of compatibility problems if the protocol must be backward compatible. UTF8 encoding is used to convert wide UNICODE characters into a string compatible with existing IRC servers and clients.

To permit the full range of UNICODE characters, we must introduce an additional post-processing step on the result of an UTF8 translation.

Any string beginning with a '%' character (i.e. "reason" strings within a REDIRECT command) will be interpreted as UTF8-encoded UNICODE strings.

UNICODE characters encoded in UTF8 may use more bytes than an ASCII character. To ensure compatibility, UNICODE strings such as nicknames and channel names must fit within the standard length measured in bytes, not in characters.

The quoting character for the post-processing step is the '\' character. All mappings are listed in the table below.

Table 1 - Character Quoting in UTF8

\b	" " (blank)
\c	","
\\	"\"
\r	CR
\n	LF
\t	TAB

IRCX clients view UTF8-encoded UNICODE strings in their native form. So non-IRCX clients can inter-operate with UNICODE nicks, UNICODE nicks are translated by the server into a usable form before being sent to the non-IRCX client. This usable format is a simple hex format preceded by a '^' character. Non-IRCX clients can use this hex format nickname to specify the IRCX/UNICODE user.

4. Terms and Definitions

Throughout this document we will use certain terms which are

defined in the following pages.

String lengths are indicated in "characters", referring to ASCII characters, because even UNICODE strings must be encoded in ASCII for the IRC protocol.

Table 2 - Definition of Terms

Chat Network	A Chat Network is comprised of one or more servers linked together.
Server	A server is a single machine.
Channel	A channel (sometimes called a room or conference) is a conversation between one or more users.
Member	A member is a user that is part of a conversation in a channel.
User	A user is a client that makes a connection to the server.
Objects	An object is a general term for channels, users, members (users in a channel), and servers. The type of object can be determined from the first character of the object's name.

4.1. User Access Levels

There are six user levels that define the ability of the user to perform operations. Some levels are determined on the context of the operation to be performed.

Table 3 - Definition of Client Levels

Sysop Manager	A Sysop Manager has full access to online commands.
Sysop	A Chat Sysop oversees and controls the chat network.
Channel Owner	A Channel Owner manages a channel and the channel hosts.
Channel Host	A Channel Host manages a channel. Also referred to as a channel operator.
Channel Member	A Channel Member is a member of a channel.
Chat User	A Chat User is a client connected to the server.

4.2. Prefixes

Each object contains a unique prefix that identifies the type of object. By tagging UNICODE objects with a special prefix, a client can easily decide whether to use standard ASCII or UNICODE when sending a message to a user or channel. The IRCX client is responsible, when sending a UNICODE string, for

prefixing it with a '%' character so that other IRCX clients can interpret it as UNICODE.

Table 4 - Object identifiers

#	The '#' prefix identifies a RFC1459 global channel.
&	The '&' prefix identifies a RFC1459 local channel.
%#	The "%#" prefix identifies an extended global channel name (a modified UTF8-encoded UNICODE string).
%&	The "%&" prefix identifies an extended local channel name (a modified UTF8-encoded UNICODE string).
%	The '%' character followed by a space or comma can identify the last channel that this client specified and is a member of. Its function is to optimize server processing of multiple messages from a client to a channel.
A to }	The 'A' through '}' prefix identifies a standard RFC1459 nickname.
'	The ''' prefix identifies an extended IRCX nickname which consists of a modified UTF8-encoded UNICODE string. The ''' character followed by a space or comma is used to represent the local client connection. The characters following ''' can be '0' through '9', '-', and any character above 'A'.
^	The '^' prefix identifies a nickname which was translated from UTF8 into hex characters so that the nickname can be viewed by non-IRCX clients. The characters following '^' can be any standard RFC1459 nickname characters.
0	The '0' prefix identifies an internal object identifier (OID). The OID consists of the '0' prefix and eight hexadecimal characters. If not supported by the server, then OID must be returned as '0'.
$	The '$' prefix identifies a server on the network. The '$' character followed by a space or comma may be used to represent the local server the client is connected to.

5. IRCX Client Messages

This section details commands which have been added and commands which have been changed from RFC1459. Responses from the server are discussed in greater detail in later sections.

5.1. ACCESS command (new IRCX command)

Create an "access entry" for an object to grant or deny access to an object, including a channel, a user, or the server. ACCESS LIST is used to list all access entries for an object and CLEAR is used to clear all entries or all entries of the same level.

Syntax 1: ACCESS <object> LIST

Syntax 2: ACCESS <object> ADD|DELETE <level> <mask> [<timeout> [:<reason>]]

Syntax 3: ACCESS <object> CLEAR [<level>]

5.1.1. Parameters

<object> The object to which access is being granted or limited. Can be a channel name, nickname, $ or *.

<level> The level of access being given or taken away. Can be:

DENY	Disallow access to an object that is accessible
GRANT	Allow access to an object that is inaccessible
HOST	Channel host access to specified channel
OWNER	Channel owner access to specified channel
VOICE	Voice access to specified channel

<mask> Mask to identify user by nickname, userid, host or server characteristics. Supports wildcards * and ?.

<timeout> Minutes until the access entry is removed. A value of 0 indicates unlimited duration.

<reason> Text reason shown when users are denied access due to the access entry.

5.1.2. Results

IRCRPL_ACCESSADD

IRCRPL_ACCESSDELETE

IRCRPL_ACCESSSTART

IRCRPL_ACCESSLIST

IRCRPL_ACCESSEND

5.1.3. Possible Errors

In this specification possible error messages are listed.

IRCERR_BADLEVEL

IRCERR_DUPACCESS

IRCERR_MISACCESS

IRCERR_TOOMANYACCESSES

IRCERR_TOOMANYARGUMENTS

IRCERR_BADCOMMAND

IRCERR_NOTSUPPORTED

IRCERR_NOACCESS

5.1.4. Remarks

An object with GRANT access entries but no DENY entries is assumed to be off-limits to those users not covered by the GRANT entries. If there are multiple entries in the access list, the list is processed in the following order: OWNER, HOST, VOICE, GRANT, DENY.

Hosts and Owners may add and delete access entries for their channel. Hosts may not remove access entries added by owners. Any user may add and delete access entries for themselves. Sysops and sysop managers on a server may add and delete access entries for that server or the entire network.

A user who has blocked another user does not get any messages from the blocked user, including invites.

5.1.5. Examples

To make a user, "piper", a channel host when they join the channel, "#mychan":

ACCESS #mychan ADD HOST piper

To grant normal access to the network to a few people but deny access to all others:

ACCESS * ADD DENY * :closed for private party

ACCESS * ADD GRANT *.company.com :these people can get on

To delete the DENY access record on the network that denies access to '*' (note that this doesn't delete other DENY access records such as '*.com'):

ACCESS * DELETE DENY *

To clear all DENY-level entries on the local server:

ACCESS $ CLEAR DENY

To clear all access entries of any level for a channel:

ACCESS #mychan CLEAR

5.2. AUTH Command (new IRCX command)

Authenticate the client using an SASL[4] authentication mechanism.

Syntax 1: AUTH <name> <seq> [:<parameter>]

5.2.1. Parameters

<name> The name of the SASL mechanism to use for authentication.

<seq> The sequence is a value of 'I' or 'S'. The 'I' value is specified for the initial AUTH message and the 'S' value is specified for all subsequent AUTH messages. If the client specifies '*' for the sequence, the server will abort the authentication sequence and return IRCERR_AUTHENTICATIONFAILED.

<parameter> This is optional data sent as an argument with the AUTH messages. The content depends on the authentication mechanism being used. All content must use standard escaping formats (e.g. \r for carriage return) to comply with IRC message format restrictions.

5.2.2. Results

AUTH message

5.2.3. Possible Errors

ERR_NEEDMOREPARAMS

IRCERR_ALREADYAUTHENTICATED

IRCERR_ALREADYREGISTERED

IRCERR_AUTHENTICATIONFAILED

IRCERR_AUTHENTICATIONSUSPENDED

IRCERR_BADCOMMAND

IRCERR_UNKNOWNPACKAGE

5.2.4. Remarks

If the server is known to support IRCX with specific SASL mechanism(s), then the first message from an IRCX-compliant client must be the AUTH command (if authentication is to be used), before the USER and NICK commands. Use MODE ISIRCX to determine if the server supports IRCX.

5.2.5. Example

Client: AUTH NTLM I :<data>

Server: AUTH NTLM S :<data>

Client: AUTH NTLM S :<data>

Server: AUTH NTLM * userid@domain 03FA4534C

5.3. CREATE (new IRCX command)

Create a new channel and/or join an existing channel.

Syntax: CREATE <channel> [<modes> [<modeargs>]]

5.3.1. Parameters

<channel> The name of the channel.

<modes> Initial channel modes, not separated by spaces (like MODE command). Includes mode 'e' to force a clone of a clonable channel.

<modeargs> Optional mode arguments, separated by spaces, in the same order as the modes. For the mode 'l' the mode argument is the maximum number of members in the channel. For the mode 'k' the mode argument is the channel keyword. These are the only modes that require mode arguments.

5.3.2. Results

CREATE message

JOIN message

RPL_TOPIC

RPL_NAMEPLY

RPL_ENDOFNAMES

5.3.3. Possible Errors

IRCERR_CHANNELEXIST

IRCERR_ALREADYONCHANNEL

ERR_NEEDMOREPARAMS

ERR_INVITEONLYCHAN

ERR_CHANNELISFULL

ERR_BANNEDFROMCHAN

ERR_BADCHANNELKEY

ERR_TOOMANYCHANNELS

ERR_UNKNOWNCOMMAND

5.3.4. Remarks

The CREATE command provides a method to specify channel properties at creation time, and also provides a method (flag 'c') to create and join a channel only if it does not already exist. If user is not in IRCX mode, server returns ERR_UNKNOWNCOMMAND.

5.3.5. Examples

This example shows the creation of a moderated (m) channel with the following properties and modes: its topic can only be changed by an owner or host (t = TOPICOP), messages from outside the channel are blocked (n = NOEXTERN), it is limited to 50 members (l 50), it has a member key which is 'password' (k password), and it will be created only if it doesn't already exist (c = CREATE)

Client: CREATE #MyChannel tnmlkc 50 password

Server: CREATE #MyChannel 048532944

5.4. DATA / REQUEST / REPLY (new IRCX command)

Used to send tagged data, requests or replies. The syntax for each is the same, but clients can use REQUEST to indicate that a REPLY is desired, and use REPLY to indicate that a REQUEST was issued. If user is not in IRCX mode, server returns ERR_UNKNOWNCOMMAND.

Syntax 1: DATA <target> <tag> :<message>

5.4.1. Parameters

<target> The target for the data. Target can be a nick, list
     of nicks, channel, list of channels, or channel followed
     by a list of some members of the channel. This

definition for <target> will be used throughout this document.

<tag> Text field that clients use to know how to interpret the data. Valid characters are [A..z], [0..9] and period (.). The first character must be one of [A..z]. Maximum 15 characters. If the tag begins with ADM, SYS, OWN or HST then the originator must have appropriate privileges (Sysop Manager, Sysop, Channel Owner or Channel Host. Channel Owner & Host apply to the channel the message is being sent to.) The server itself can send tags beginning with these reserved strings.

<message> Payload or data for the message, ending with a newline. Any newlines or other control characters within the body of the message must be escaped.

5.4.2. Results

DATA message

5.4.3. Possible Errors

ERR_UNKNOWNCOMMAND

5.4.4. Remarks

REPLY and REQUEST may be used to replace "DATA". IRCX servers should not send DATA commands to clients that have not indicated that they support IRCX. Clients should only process DATA messages if they know and support the tag used.

Prefixes should indicate the organization that specified them, when appropriate. For example: MYORG.AVATAR could be a tag used by MyOrg, as would be all MYORG.* tags.

5.4.5. Example

Sending ad banners from server sysop to channel, "#Channel":

DATA #Channel SYS.AD.SMALL :<url-stuff>

5.5. EVENT (new IRCX command)

Add/Change/Delete event logging to the client connection. The list of event types and the way masks are applied is not specified here.

Syntax 1: EVENT [ADD | DELETE] <event> [<mask>]

Syntax 2: EVENT LIST [<event>]

5.5.1. Parameters

<event> Type of event, such as CHANNEL, MEMBER, SERVER, CONNECTION, SOCKET or USER.

<mask> Optional mask for applying a selection criteria per event.

5.5.2. Results

IRCRPL_EVENTADD

IRCRPL_EVENTDEL

IRCRPL_EVENTSTART

IRCRPL_EVENTLIST

IRCRPL_EVENTEND

5.5.3. Possible Errors

ERR_NEEDMOREPARAMS

ERR_NOPRIVILEGES

IRCERR_BADFUNCTION

IRCERR_EVENTDUP

IRCERR_EVENTMIS

IRCERR_NOSUCHEVENT

IRCERR_TOOMANYEVENTS

5.5.4. Remarks

The EVENT command can be used by other server implementations to define the events that a sysop manager or sysop of the server wishes to be notified of. Only sysop managers and/or sysops are allowed to receive event messages.

5.5.5. Examples

Add channel events.

Client: EVENT ADD CHANNEL

Server: :<host> 806 <nick> CHANNEL *!*@*$*

Add a list event with no active events returned.

Client: EVENT LIST

Server: :<host> 808 <nick> :Start of events

:<host> 809 <nick> CHANNEL *!*@*$*

:<host> 810 <nick> :End of events

5.6. IRCX (new IRCX command)

Enables IRCX mode and displays IRCX status. Use MODE ISIRCX first to see if the server supports IRCX.

Syntax: IRCX

5.6.1. Parameters

None.

5.6.2. Results

IRCRPL_IRCX

5.6.3. Example

This example includes a server that supports IRCX and three authentication packages.

Client: IRCX

Server: :<host> 800 <nick> 1 0 NTLM,DPA,ANON 512 *

5.7. ISIRCX (new IRCX command)

Queries whether or not the server supports the Extended RFC1459 protocol (IRCX). Server response is same as for IRCX message.

Use ISIRCX after logging into the server.

Syntax: ISIRCX

5.7.1. Results

IRCRPL_IRCX

5.8. LISTX (new IRCX command)

Extended version of the LIST command that returns additional channel properties. Channels with extended names will be returned, even to non-IRCX clients. Some channel modes and the PICS rating string are included with the result. Only the channel modes that do not define an additional argument (TOPICOP, NOEXTERN, etc.) are included in the <modes> field.

Clients should use the query limit to avoid overloading the client or having the connection dropped by the server.

Syntax 1: LISTX [<channel list>]

Syntax 2: LISTX <query list> [<query limit>]

5.8.1. Parameters

<channel list> A list of channels may be specified in order to find the PICS ratings or modes of those channels. If no channels are specified, the server will send the entire matching list of channels.

<query list> One or more query terms separated by spaces or commas.

Table 5 - Query terms for LIST command

<#	Select channels with less than # members.
>#	Select channels with more than # members.
C<#	Select channels created less than # minutes ago.
C>#	Select channels created greater than # minutes ago.
L=<mask>	Select channels with language property matching the mask string.
N=<mask>	Select channels with name matching the mask string.
R=0	Select unregistered channels.
R=1	Select registered channels.
S=<mask>	Select channels with subject matching the mask string.
T<#	Select channels with a topic changed less than # minutes ago.
T>#	Select channels with a topic changed greater than # minutes ago.
T=<mask>	Select channels that topic matches the mask string.
<query limit>	Maximum number of channels to be returned.
<mask>	Sequence of characters that is used to select

a matching channel name or topic. The character * and ? are used for wildcard searches.

5.8.2. Results

IRCRPL_LISTXSTART

IRCRPL_LISTXLIST

IRCRPL_LISTXPICS

IRCRPL_LISTXTRUNC

IRCRPL_LISTXEND

5.8.3. Remarks

To compose a mask, use this character escaping scheme.

Table 6 - Character escaping in mask-string

\b for " " blank

\c for ","

\\ for "\"

* for *

\? for ?

The PICS property is only returned if not null.

A record limit of '0' (zero) or blank will be interpreted as unlimited.

5.9. MODE (extension to RFC1459 command)

MODE command can now be used for new user or channel modes (see later sections). In addition, the special syntax "MODE ISIRCX" can be used to help clients find out whether a server supports IRCX prior to logging into the server.

Syntax: MODE ISIRCX

5.9.1. Results

MODE message

IRCRPL_IRCX

5.9.2. Remarks

The ISIRCX mode (must be in capital letters) is only functional before the user has registered with the server.

IRCX servers respond with IRCRPL_IRCX and non-IRCX servers should return an error code. This allows unregistered users to query whether the server supports IRCX.

See RFC1459 for more details on the original MODE command and its parameters.

5.10. NOTICE (extension to RFC1459 command)

Sends a notice to a channel or user. In RFC1459 a notice could only be sent to a channel or a list of users. Now a notice can be sent to a list of users within the context of a channel. As before, users will not get the list of users that received the notice.

Syntax: NOTICE <target> :<message>

5.10.1. Results

Same as defined in RFC1459, but with the additional functionality of sending to a list of nicknames within the context of a channel.

5.11. PRIVMSG (extension to RFC1459 command)

Sends a message to a channel or user. PRIVMSG is extended in the same way as NOTICE.

Syntax: PRIVMSG <target> :<message>

5.11.1. Results

Same as defined in RFC1459, but with the additional functionality of sending to a list of nicknames within the context of a channel.

5.12. PROP (new IRCX command)

Add, change or delete a channel data property.

To query a property:

Syntax 1: PROP <channel> <property>[,<property>]

To set or change a property:

Syntax 2: PROP <channel> <property> :<data>

To delete a property:

Syntax 3: PROP <channel> <property> :

5.12.1. Results

PROP message

IRCRPL_PROPLIST

IRCRPL_PROPEND

5.12.2. Possible Errors

IRCERR_BADCOMMAND

IRCERR_BADPROPERTY

IRCERR_SECURITY

IRCERR_NOSUCHOBJECT

IRCERR_TOOMANYARGUMENTS

IRCERR_BADVALUE

5.12.3. Remarks

The PROP command provides a new method for manipulating string and numeric properties of channels. If an optional channel property is not supported on the server, the server should return IRCERR_BADPROPERTY.

See section 8.2 for definitions of channel properties.

5.12.4. Examples

Client: PROP #MyChan TOPIC,ONJOIN

Server: :<host> 818 <nick> #MyChan TOPIC :This is the topic of my channel

Server: :<host> 818 <nick> #MyChan ONJOIN :Welcome to my channel!

Server: :<host> 819 <nick> #MyChan :End of properties

Client: PROP #MyChannel TOPIC :Change my channel topic

Server: PROP #MyChannel TOPIC :Change my channel topic

5.13. WHISPER (new IRCX command)

Whisper to member(s) in a channel.

Syntax: WHISPER <channel> <nick-list> :<message>

5.13.1. Results

WHISPER message

5.13.2. Possible Errors

ERR_UNKNOWNCOMMAND

ERR_CANNOTSENDTOCHAN

ERR_USERNOTINCHANNEL

ERR_NEEDMOREPARAMS

IRCERR_NOTONCHANNEL

IRCERR_TOOMANYTARGETS

IRCERR_NOSUCHNICK

IRCERR_NOSUCHCHANNEL

IRCERR_NOWHISPER

5.13.3. Remarks

The purpose of the WHISPER command is to whisper to one or more members in a channel within the context of that channel. The sender and all recipients must be in the channel specified. A whisper is different from a privmsg in that it includes both a user list and a channel name, indicating that the message is private but has a context.

IRCX clients should display the WHISPER message within the context (possibly a window) of the specified channel. Non-IRCX clients receive a PRIVMSG with the content of the whisper (losing the context of the whisper). Only one whisper will be sent per nick. A user may whisper to themselves.

The channel mode 'w' will disable whispers between ordinary members of the channel (but not whispers from or to channel operators).

6. IRCX Server Messages

This section summarizes new extended messages which can be sent from the server.

6.1. AUTH (new IRCX message)

Negotiates authentication with client or informs client that authorization was successful.

Syntax 1 (negotiating authorization): AUTH <name> S [:<parameter>]

Syntax 2 (authorization successful): AUTH <name> * <ident>
<oid>

6.1.1. Parameters

<name> The name of the SASL mechanism to use for authentication.

<ident> The ident is the userid@domain of the authenticated client account. It is returned on the final response from the server (Syntax 2) when authentication is successful.

<oid> The OID is the internal object identifier assigned to the client connection. If not supported by the server, then OID must be returned as '0'.

<parameter> This is optional data sent as an argument with the AUTH messages.

6.1.2. Remarks

The server will send the syntax 2 form when the authentication process is complete or a numeric error reply.

6.2. CLONE (new IRCX message)

Informs the hosts and owners in a CLONEABLE channel that a new CLONE channel was created.

Syntax: CLONE <channel-name> <oid>

6.2.1. Parameters

<channel-name> The name of the created CLONE channel.

<oid> The object identifier for this new CLONE channel. The value is implementation-dependent and must equal 0 if not supported.

6.2.2. Remarks

The CLONE message can be sent at anytime to the hosts/owners of a CLONEABLE channel to inform them that a new CLONE channel was created. This message is intended to be used by a custom application to automatically join the new CLONE channel. A non-IRCX client will not get this message.

6.2.3. Example

Server: CLONE #MyChat1 034F8FF32

6.3. CREATE (new IRCX message)

Informs the client that a new channel was created. Response

to the CREATE command.

Syntax: CREATE <channel-name> <oid>

6.3.1. Parameters

<channel-name> The name of the created channel.

<oid> The object identifier for this new channel. The value is implementation- dependent and must equal 0 if not supported.

6.3.2. Remarks

The CREATE message is sent in response to a CREATE command sent by the client application if the channel specified did not previously exist and was created by the server.

6.3.3. Example

Server: CREATE #MyChat1 034F8FF32

6.4. DATA / REQUEST / REPLY (new IRCX messages)

The DATA message (could be REQUEST or REPLY also) is forwarded from another user or sent by the server. The payload or message should be interpreted according to the tag. If the tag is unknown, the message may be discarded.

Syntax 1: <sender> :DATA <target> <tag> :<message>

<sender> :REQUEST <target> <tag> :<message>

<sender> :REPLY <target> <tag> :<message>

If the DATA, REQUEST or REPLY message is sent to a number of members within a channel, the receiving user will see the channel name and their own nick in the message as follows:

Syntax 2: <sender> :DATA <channel> <nick> <tag> :<message>

6.4.1. Parameters

<sender> May be a user or a server.

<target> Channel and/or nick list as defined above.

<tag> Identifying tag.

<message> Payload.

6.4.2. Remarks

The tag indicates what to do with the message. Tag types may be specified by administrators, client developers, server developers etc.

A tag beginning with SYS can only be from a sysop, sysop manager or the server. A tag beginning with ADM can only be from a sysop manager or the server.

DATA message functionality is different from client-to-client messaging in several respects. First, we encourage groups to define their own tags and data formats for special purposes, for example to indicate details for a user's avatar in graphical chats, or to indicate that an ad banner or image should be downloaded. Second, the DATA message is more appropriate for content that may go to several users, for example all users in a channel. Third, the DATA message may come from the server. Fourth, the SYS and ADM prefixes are specified so that important tags may be reserved for sysops, sysop managers and the server itself, with the server responsible for verifying the sender before forwarding the DATA message.

6.5. EVENT (new IRCX message)

Notifies the client of an event. These events are intended for sysops and sysop managers, and are sent in addition to IRC events.

Syntax: EVENT <time-stamp> <object> <event type> <parameters>

6.5.1. Parameters

<time-stamp> The number of elapsed seconds from midnight (00:00:00) January 1, 1970 (coordinated universal time) until the time that the event occurred on the server.

<object> Objects can be Channel, Member, User, Connection, Socket or Server.

<event type> Event type varies depending on the object. For example, events for channels can be Create, Destroy, Topic change, Mode change, Collision.

<parameters> Vary depending on event type.

6.5.2. Example

This example is the event generated when a user logs onto server, "chat1" with the nickname, "john", showing the user's info including IP address and port.

EVENT chat1 946080000 USER LOGON john!jsmith@uw.edu 192.29.93.93:1111

6.6. KNOCK (new IRCX message)

Informs the owners and hosts of a channel that a user has

tried to enter the channel and could not (could be because of a full channel, keyword wrong, user ban, or other reasons). This message is only sent to the IRCX owners and hosts of the channel.

Syntax: <user> KNOCK <channel> <reason>

6.6.1. Parameters

<user> User field is of the format nick!userid@host.

<channel> Name of the channel that the user tried to enter.

<reason> Reason that the user received when they tried to join the channel.

6.6.2. Remarks

A KNOCK message will not be sent to a non-IRCX client.

6.7. REDIRECT (new IRCX message)

Informs the client that they need to connect to another server.

Syntax: REDIRECT <server-list> :<reason>

6.7.1. Parameters

<server-list> The server list is a comma separated list of host:port pairs. The server list can be specified either as a fully-qualified domain name or by the IP address in quad-dotted notation.

<reason> The redirect reason is an implementation-dependent string which can optionally be displayed to the client.

6.7.2. Remarks

The REDIRECT message can be sent to the client at any time to request that the client disconnect and reconnect to another server specified in the list. The REDIRECT message is generally sent when a server is to be shutdown.

A REDIRECT message will not be sent to a non-IRCX client.

6.7.3. Example

Server: REDIRECT chat.corp.net:6667,134.9.3.3:6667 :Server full.

6.8. WHISPER (new IRCX message)

A whisper from another channel member.

Syntax: <sender> WHISPER <channel> <nick list> :<text>

6.8.1. Parameters

<sender> The nick-name of the person that sent the whisper.

<channel> The channel that the message is sent to.

<nick list> The list of nicknames of channel members that will receive the whisper.

<text> The content of the whisper.

6.8.2. Remarks

The server may transform a WHISPER into a PRIVMSG for clients that do not support IRCX.

6.8.3. Example

Server: Joe WHISPER #test Jill :Test whisper.

7. User Modes and Properties

These are new user modes and properties that have been added. The MODE command as defined in RFC1459 is used to add or remove modes.

7.1. OWNER (IRCX +q)

The OWNER mode indicates that the user is an owner of a channel. Only a channel owner can give OWNER mode to another member of that channel, but any owner may remove OWNER mode from themselves.

Clients in IRCX mode see owner nicknames with a '.' prefix and continue to see hosts with a '@' prefix (in results from NAMES, WHO, WHOIS and other commands which list nicknames).

Note that HOST mode uses +o, which was "operator" mode in RFC1459. Syntax for these modes is the same as "operator" mode:

MODE <channel> { + | - }q <nick>

7.2. GAG (IRCX +z)

The GAG mode is applied by a sysop or sysop manager on a user and may not be removed except by a sysop or sysop manager. The user may not be notified when this mode is applied because the mode can be a more effective tool for keeping order if the user doesn't know exactly when it is applied.

The server will discard all messages from a user with GAG mode to any other user or to any channel.

MODE <nick> { + | - }z

8. Channel Modes and Properties

New modes and properties have been added to channels.

Channels support four mutually exclusive states of visibility: public, private, hidden, and secret. The visibility of a channel affects which modes and properties are available to a client.

8.1. Modes

Each channel object contains a number of binary mode settings that can be queried and optionally updated via the RFC1459 MODE command.

8.1.1. PUBLIC (RFC1459 default)

The channel is public and all information about the channel (except for text messages) can be queried by non-members. PUBLIC mode is mutually exclusive with the PRIVATE, HIDDEN and SECRET modes.

This mode may be set and queried by sysop managers, owners and hosts of the channel. It may be queried by sysops, members of the channel, and users.

8.1.2. PRIVATE (RFC1459 +p)

The channel is private and only the name, number of members and PICS property can be queried by non-members. PRIVATE mode is mutually exclusive with the PUBLIC, HIDDEN and SECRET modes.

This mode may be set by sysop managers, owners and hosts of the channel. It may be not be queried by sysops or users.

8.1.3. HIDDEN (IRCX +h)

The channel is hidden and cannot be located by enumeration queries from non-members. HIDDEN mode is mutually exclusive with the PUBLIC, PRIVATE, and SECRET modes. The purpose of the new HIDDEN channel mode is to permit the existence of channels that cannot be found using the standard LIST command, but whose properties can be queried if the exact channel name is known. Thus a HIDDEN channel is the same as a PUBLIC channel except that it cannot be enumerated by using a LIST or LISTX command.

This mode may be set and queried like PUBLIC mode.

8.1.4. SECRET (RFC1459 +s)

The channel is secret and cannot by located by any query from non-members. SECRET mode is mutually exclusive with the PUBLIC, PRIVATE, and HIDDEN modes.

This mode may be set and queried like PRIVATE mode.

8.1.5. MODERATED (RFC1459 +m)

Normally, new channel members may speak. In MODERATED mode however, new channel members may not speak by default. This is achieved by giving all new members the '-v' mode (no voice) for that channel. Usually -v mode has no effect, but in a MODERATED channel it does.

This mode may be set and queried by sysop managers, owners and hosts of the channel. It may be queried by sysops and members of the channel. Users may query this mode if the channel is PUBLIC or HIDDEN.

8.1.6. NOEXTERN (RFC1459 +n)

NOEXTERN mode blocks messages from non-members to the channel. A sysop manager can still send a message to the channel.

This mode may be set and queried like MODERATED mode.

8.1.7. TOPICOP (RFC1459 +t)

TOPICOP mode permits only channel owners and hosts to change the channel topic property.

This mode may be set and queried like MODERATED mode.

8.1.8. INVITE (RFC1459 +i)

INVITE mode permits only invited users to enter the channel. Only owners and hosts can issue an invitation when this mode is on.

This mode may be set and queried like MODERATED mode.

8.1.9. KNOCK (IRCX +u)

The KNOCK extended mode causes a KNOCK message to be sent to owners and hosts of the channel if a user attempts to join a channel and the server denies entrance.

This mode may be set and queried like MODERATED mode.

8.1.10. NOFORMAT (IRCX +f)

The NOFORMAT channel mode is an indication to the client application not to format text received from the channel.

Normally clients will prefix text messages with "x said y" or "x whispers to y and x", if the NOFORMAT mode is set then just the raw text should be displayed moving to the next line at the end of the message. The client should not echo text sent by the client. This is to permit custom applications to control the formatting to clients. Clients may want to inform users that messages can be spoofed with this mode.

This mode can only be set by sysop managers. It can be read by sysop managers, sysops, owners, hosts and members of the channel. It can be read by users if the channel is PUBLIC or HIDDEN.

8.1.11. NOWHISPER (IRCX +w)

The NOWHISPER channel mode will prevent all messages sent to specific nicknames within the context of that channel.

This channel mode may be set and read by sysop managers and owners. It can be read by sysops, hosts and members of the channel. It can be read by other users if the channel is PUBLIC or HIDDEN.

8.1.12. AUDITORIUM (IRCX +x)

The AUDITORIUM channel mode restricts visibility and messaging within a channel. Members will only see themselves and the hosts/owners in the channel. Any message sent by a member will only be received by the hosts and owners. Hosts and owners will see all members in the channel and messages from a host or owner are seen by all channel members. Ordinary members will not receive JOIN/PART messages from other members. This mode is designed for channels with so many members that they do not want to see each other. Note that auditorium mode may only be set at channel creation time using the CREATE command.

This mode may be set and read by sysop managers, sysops, and owners. It may be read by hosts and members of the channel. It can be read by other users if the channel is PUBLIC or HIDDEN.

8.1.13. REGISTERED (IRCX +r)

The channel is registered by the administrator of the chat network. The registration procedure is not defined here. Only the server or server administrator can set this mode.

It can be read by sysop managers, sysops, owners, hosts, and

members of the channel. It can be read by users if the channel is PUBLIC or HIDDEN.

8.1.14. SERVICE (IRCX +z)

A service is monitoring/controlling the channel. This is an indication to the client that message traffic sent to the channel is being monitored by a Chat Service which does not appear as a member in the channel.

This mode can only be set by the Chat Server. It can be read by sysop managers, sysops, owners, hosts and members of the channel. It can be read by users if the channel is PUBLIC or HIDDEN.

8.1.15. AUTHONLY (IRCX +a)

The AUTHONLY channel mode permits channel access only to users who have been authenticated by the server. Note that an authenticated user is any user that was successfully authenticated with the PASS command or the AUTH command.

This mode can be set and read by sysop managers or owners of the channel. It can be read by sysops, hosts and members of the channel. It can be read by users if the channel is PUBLIC or HIDDEN.

8.1.16. CLONEABLE (IRCX +d)

The CLONEABLE channel mode defines a channel that creates new clone channels if the parent channel is full. A clone channel is created with the same name as the parent cloneable channel with a numeric suffix starting at 1, ranging to 99. It is not valid to set the CLONEABLE channel mode of a parent channel that ends with a numeric character. The clone channel inherits modes and properties from the parent channel when it is set up. When a clone channel is created, any channel that has the same name is removed. This prevents channel takeover of a clone channel.

It is advised that only sysop be allowed to set cloneable mode on a channel. The mode may be read by sysops, owners, hosts and members of the channel. It may be read by users if the channel is PUBLIC or HIDDEN.

8.1.17. CLONE (IRCX +e)

The CLONE channel mode defines a channel that was created by the server when a parent CLONEABLE channel becomes full. Users should usually join the parent channel, although a user may join a clone channel that is not full. A sysop manager can set up a clone channel but only when the channel is being created.

This mode can be set by the sysop manager and read like the SERVICE mode.

8.2. Properties

Each channel object contains a number of properties that can be queried and optionally updated via the IRCX PROP command. Owners and hosts may update a property for their own channel. Sysop Managers and Sysops can use the PROP command on a channel only by becoming host/owner of that channel. Users and members can read properties only.

8.2.1. OID (R/O)

The OID channel property is the internal object identifier for the channel. As a shortcut, the OID can be optionally used in place of the full string name of a channel. If the OID is set to "0", then this feature is not supported on the server.

This property may be read by all users, except by ordinary users when the channel is SECRET or PRIVATE.

8.2.2. NAME (R/O)

The NAME channel property is the name of the channel (limited to 63 characters, including 1 or 2 characters for channel prefix). Valid characters are as defined in RFC1459.

This property may be set and read like the OID property.

8.2.3. CREATION (R/O)

The CREATION channel property is the time that the channel was created, in number of seconds elapsed since midnight (00:00:00), January 1, 1970, (coordinated universal time).

This property may be set and read like the OID property.

8.2.4. LANGUAGE

The LANGUAGE channel property is the preferred language type. The LANGUAGE property is a string limited to 31 characters. We recommend following the guidelines of RFC1766[6] to form well-understood language-defining strings.

This property may be set and read by sysop managers, owners and hosts of the channel. It may be read by sysop managers, sysops and members. It may be read by users if the channel is PUBLIC or HIDDEN.

8.2.5. OWNERKEY

The OWNERKEY channel property is the owner keyword that will provide owner access when entering the channel. The OWNERKEY property is limited to 31 characters.

This property may be set by the sysop manager or channel owner. It may never be read.

8.2.6. HOSTKEY

The HOSTKEY channel property is the host keyword that will provide host (channel op) access when entering the channel. The HOSTKEY property is limited to 31 characters.

This property may be set and read like the OWNERKEY property.

8.2.7. MEMBERKEY

The MEMBERKEY channel property is the keyword required to enter the channel. The MEMBERKEY property is limited to 31 characters. This is backwards-compatible with RFC1459 because users can still use the MODE command as an alternative way to set this property.

This property may be set and read like the OWNERKEY property.

8.2.8. PICS

The PICS channel property is the current PICS rating of the channel. Chat clients that are PICS enabled can use this property to determine if the channel is appropriate for the user. The PICS property is limited to 255 characters. The format for this field is defined by PICS (see http://www.w3.org).

This property may be set by sysop managers and read by all. It may not be read by ordinary users if the channel is SECRET.

8.2.9. TOPIC

The TOPIC channel property is the current topic of the channel. The TOPIC property is limited to 160 characters.

This property may be set and read by sysop managers, owners and hosts of the channel. It may be read by sysops and members of the channel. It may be read by users if the channel is PUBLIC or HIDDEN.

8.2.10. SUBJECT

The SUBJECT channel property is a string that can contain subject keywords for search engines. The SUBJECT property is limited to 31 characters.

This property may be set and read like the TOPIC property.

8.2.11. CLIENT

The CLIENT channel property contains client-specified information. The format is not defined by the server. The CLIENT property is limited to 255 characters.

This property may be set and read like the TOPIC property.

8.2.12. ONJOIN

The ONJOIN channel property contains a string to be sent (via PRIVMSG) to a user after the user has joined the channel. The channel name is displayed as the sender of the message. Only the user joining the channel will see this message. Multiple lines can be generated by embedding '\n' in the string. The ONJOIN property is limited to 255 characters.

This property may be set and read by sysop managers, owners and hosts of the channel. It may additionally be read by sysops.

8.2.13. ONPART

The ONPART channel property contains a string that is sent (via NOTICE) to a user after they have parted from the channel. The channel name is displayed as the sender of the message. Only the user parting the channel will see this message. Multiple lines can be generated by embedding '\n' in the string. The ONPART property is limited to 255 characters.

This property may be set and read like the ONJOIN property.

8.2.14. LAG

The LAG channel property contains a numeric value between 0 to 2 seconds. The server will add an artificial delay of that length between subsequent messages from the same member. All messages to the channel are affected.

This property may be set and read by sysop managers and owners. It can additionally be read by sysops, hosts, and members of the channel. It can be read by users if the channel is PUBLIC or HIDDEN.

8.2.15. ACCOUNT

The ACCOUNT channel property contains an implementation-dependent string for attaching a security account. This controls access to the channel using the native OS security system. The ACCOUNT property is limited to 31 characters.

This property may be set by sysop managers. It can only be read by sysop managers, sysops and owners of the channel.

8.2.16. CLIENTGUID

The CLIENTGUID channel property contains a GUID that defines the client protocol to be used within the channel.

This property may be set and read like the LAG property.

8.2.17. SERVICEPATH

The SERVICEPATH channel property contains the path of a server side extension that is used to control the channel operation. Details are implementation-dependent.

This property may be set and read like the LAG property.

9. IRCX Command Responses

The new extended IRCX numeric replies follow the same convention as IRC replies, with a specific range for command responses and another range for error results. The IRCX command responses are in the range of 800 to 899 and 900 to 999 for the error results.

9.1. Command Replies

800 - IRCRPL_IRCX

<state> <version> <package-list> <maxmsg> <option-list>

The response to the IRCX and ISIRCX commands. The <state> indicates if the client has IRCX mode enabled (0 for disabled, 1 for enabled). The <version> is the version of the IRCX protocol starting at 0. The <package-list> contains a list of authentication packages supported by the server. The package name of "ANON" is reserved to indicate that anonymous connections are permitted. The <maxmsg> defines the maximum message size permitted, with the standard being 512. The <option-list> contains a list of options supported by the server; these are implementation-dependent. If no options are available, the '*' character is used.

801 - IRCRPL_ACCESSADD

<object> <level> <mask> <timeout> <user> :<reason>

Response to a successful ACCESS ADD command. The <object> is the name of the object to which the access restrictions apply (i.e. channel name or user name). The <level> is the level of access being added (i.e. GRANT, DENY). The <mask> is the selection mask. If no mask is provided in the ACCESS command, then the default mask of *!*@*$* is used. The <timeout> is

the amount of time this access entry will last. The <user> is the one who added the new ACCESS record. The <reason> is the reason supplied in the ACCESS ADD command.

802 - IRCRPL_ACCESSDELETE

<object> <level> <mask>

Response to a successful ACCESS DELETE command. See reply 801 for explanation of the fields.

803 - IRCRPL_ACCESSSTART

<object> :Start of access entries

Beginning of a list of access entries. <object> is the object to which the access restrictions apply (i.e. channel name or user name). The next message will be an IRCRPL_ACCESSLIST or IRCRPL_ACCESSEND reply.

804 - IRCRPL_ACCESSLIST

<object> <level> <mask> <timeout> <user> :<reason>

One entry in a list of access entries. See reply 801 for explanation of the fields.

805 - IRCRPL_ACCESSEND

<object> :End of access entries

End of a list of access entries. See reply 803 for explanation of the field. This reply will always follow an IRCRPL_ACCESSSTART or IRCRPL_ACCESSLIST reply.

806 - IRCRPL_EVENTADD

<event> <mask>

The acknowledgment response to the EVENT ADD command. The <event> contains the name of the event added. The <mask> is the selection mask. If no mask is provided in the EVENT command, then the default mask of *!*@*$* is used.

807 - IRCRPL_EVENTDEL

<event> <mask>

The acknowledgment response to the EVENT DELETE command. The <event> contains the name of the deleted event. The <mask> is the selection mask. If no mask is provided in the EVENT command, then the default mask of *!*@*$* is used.

808 - IRCRPL_EVENTSTART

```
:Start of events
```

Response to the EVENT LIST <event> message that indicates the start of the event list.

809 - IRCRPL_EVENTLIST

<event> <mask>

Response to the EVENT LIST <event> message that displays a list of current events that the client is interested in.

810 - IRCRPL_EVENTEND

:End of events

Response to the EVENT LIST <event> message that indicates the end of the event list.

811 - IRCRPL_LISTXSTART

:Start of ListX

First reply to a LISTX (extended list) command. Will always be followed by a reply of type 812, 816 or 817.

812 - IRCRPL_LISTXLIST

<channel> <modes> <members> <member limit> :<topic>

Single list item in an extended list of channels. The <channel> is the name of the channel in the list. The <modes> specify the current modes set on the channel. <members> lists the members currently in the channel. <member limit> returns the member limit of the channel. <topic> returns the topic of the channel.

813 - IRCRPL_LISTXPICS

:<PICS-rating>

PICS rating string for the last channel listed (follows an 812 message).

816 - IRCRPL_LISTXTRUNC

:Truncation of ListX

Last reply to a LISTX command, either because the user asked for a limited list of channels or because the server truncated the list to prevent output flooding. Always follows a reply of type 811, 812 or 813.

817 - IRCRPL_LISTXEND

:End of ListX

Last reply to a LISTX command, indicating that the list has ended. Always follows a reply of type 811, 812 or 813.

818 - IRCRPL_PROPLIST

<object> <property> :<value>

A value in a property list. The <object> is the name of the object (i.e., channel name). The <property> is the property in the list. The <value> is the value of the property listed.

819 - IRCRPL_PROPEND

<object> :End of properties

Last message in a property list.

9.2. IRCX Error Replies.

900 - IRCERR_BADCOMMAND

<command> :Bad command

Response to an incorrectly formatted command.

901 - IRCERR_TOOMANYARGUMENTS

<command> :Too many arguments

Response to too many arguments being provided for a command.

902 - IRCERR_BADFUNCTION

<command> :Bad function

Response to a command that supports functions, with invalid function sent by the user. For example, the EVENT command supports functions.

903 - IRCERR_BADLEVEL

<command> :Bad level

Response to an ACCESS command with a bad level specified (i.e. not GRANT, DENY...)

904 - IRCERR_BADTAG

<command> :Bad message tag.

Response to a DATA/REQUEST/REPLY with an incorrect message tag.

905 - IRCERR_BADPROPERTY

<channel> :Bad property specified

Response to a channel property command with a bad property specified.

906 - IRCERR_BADVALUE

<channel> :Bad value specified

Response to an attempt to set an integer property to a string value (PROP command).

907 - IRCERR_RESOURCE

:Not enough resources

Server does not have enough resources to perform command.

908 - IRCERR_SECURITY

:No permissions to perform command

For security reasons, the command/function/operation is not permitted for this level of client.

909 - IRCERR_ALREADYAUTHENTICATED

<package> :Already authenticated

The client is already authenticated with the server.

910 - IRCERR_AUTHENTICATIONFAILED

<package> :Authentication failed

The authentication failed due to a bad userid/password or server/network failure.

911 - IRCERR_AUTHENTICATIONSUSPENDED

<package> :Authentication suspended for this IP

Authentication has been temporary disabled due to too many authentication failures from this IP.

912 - IRCERR_UNKNOWNPACKAGE

<package> :Unsupported authentication package

The authentication package specified is not known to the server. ISIRCX command will return a list of supported authentication packages.

913 - IRCERR_NOACCESS

<command> :No access

Response to a user trying to change the ACCESS list for an object the user does not have sufficient privileges to alter.

914 - IRCERR_DUPACCESS

:Duplicate access entry

An access entry already exists for the specified user mask.

915 - IRCERR_MISACCESS

:Unknown access entry

Response to ACCESS DELETE command when server does not recognize the entry to be removed.

916 - IRCERR_TOOMANYACCESSES

:Too many access entries

Response to ACCESS ADD command when maximum number of access entries has been reached.

918 - IRCERR_EVENTDUP

<event> <mask> :Duplicate event entry

The user has asked for an event that is already being sent.

919 - IRCERR_EVENTMIS

<event> <mask> :Unknown event entry

Response to event remove command when user is not already receiving the event specified.

920 - IRCERR_NOSUCHEVENT

<event> :No such event type

Response to event add command when server does not recognize the event specified.

921 - IRCERR_TOOMANYEVENTS

<event> :Too many events specified

Response to event add command when user may not add another event to monitor.

923 - IRCERR_NOWHISPER

<object> :Does not permit whispers

Response to whisper command when channel does not permit whispers.

924 - IRCERR_NOSUCHOBJECT

<object> :No such object found

Response to an attempt to define a property for an object which can't be found (PROP command).

925 - IRCERR_NOTSUPPORTED

<object> :Command not supported by object

Response to PROP and ACCESS commands when a bad object was specified in the command.

926 - IRCERR_CHANNELEXIST

<channel> :Channel already exists.

The channel name in the CREATE command already exists on the server and the +c mode was specified.

927 - IRCERR_ALREADYONCHANNEL

<channel> :Already in the channel.

Response to join command when user is already in the channel.

999 - IRCERR_UNKNOWNERROR

:Unknown error code <error-code>

The internal error generated doesn't map to a valid IRCX error reply. The error code is implementation-dependent.

10. Security Considerations

Security issues are also discussed in the authentication section.

The IRCX and ISIRCX commands return a set of authentication mechanisms supported by the server. This method is open to a middle man attack whereby an attacker modifies the list of returned authentication methods and only offers a clear-text password transaction. In order to avoid this type of attack, only authentication methods with a challenge response mechanism should be used.

Since all administration commands for RFC1459 and IRCX are sent in clear text, a stream layer encryption mechanism like SSL[5] or IPSEC is required to protect the integrity and confidentiality of the transactions. The mechanisms for

establishing these connection are outside the scope of this document.

11. Acknowledgements

Specific acknowledgments must be extended to the following people as the editors of the previous versions:

Kent Cedola, Lisa Dusseault, and Thomas Pfenning

In addition it has benefited from many rounds of review and comments. As so, any list of contributors is bound to be incomplete; please regard the following as only a selection from the group of people who have contributed to make this document what it is today.

In alphabetical order:

Josh Cohen, Alex Hoppman, David Kurlander, Robert Uttecht, and Teoman Smith

12. References

[1] "Internet Relay Chat Protocol", Network Working Group RFC 1459, J. Oikarinen, D. Reed, May 1993

[2] The Unicode Consortium, "The Unicode Standard Version 2.0", Addison Wesley Developers Press, July 1996

[3] "INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1", Network Working Group RFC 2060, M. Crispin, December 1996

[4] "Simple Authentication and Security Layer (SASL)", Work in Progress, draft-myers-auth-sasl-07.txt, J. Myers, December 1996

[5] "The SSL Protocol Version 3.0", Work in Progress, draft-ietf-tls-ssl-version3-00.txt, Alan O. Freier, Philip Karlton, Paul C. Kocher, November 1996

[6] "Tags for the Identification of Languages", Network Working Group RFC 1766, H. Alvestrand, March 1995

13. Authors' Addresses

Dalen Abraham

Microsoft Corporation

One Microsoft Way

Redmond, WA 98052

EMail: dalena@microsoft.com