Instant Messaging and Presence Protocol
Internet Draft
Category: Informational
Document: draft-movva-msn-messenger-protocol-00.txt
Document Expires: 2/00

R. Movva
Microsoft
August, 1999

W. Lai
Microsoft
August, 1999

MSN Messenger Service 1.0 Protocol

Status of this Memo

This document and related documents are discussed on the impp mailing list. To join the list, send mail to impp-request@iastate.edu. To contribute to the discussion, send mail to impp@iastate.edu. The archives are at http://lists.fsck.com/cgi-bin/wilma/pip. The IMPP working group charter, including the current list of group documents, can be found at http://www.ietf.org/html.charters/impp-charter.html.

1. Abstract

Microsoft released a commercial Instant Messaging product in July of 1999 called MSN Messenger Service. This document describes the protocol used by that product for core instant messaging and presence functionality. While this protocol does not meet many of the requirements of the IMPP working group, it is provided as background information on existing Instant Messaging

implementations. This protocol is provided 'as is' without warranty of any kind.

2. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119.

Protocol messages sent from client to server are preceded by "C:".

Protocol messages sent from server to client are preceded by "S:".

3. Introduction

MSN Messenger Service enables a user to learn about the presence of other people on the Internet, and to communicate with them in real-time. This functionality is commonly referred to as "Instant Messaging" (IM).

This document describes the syntax and semantics of the MSN Messenger Protocol, the communication protocol running between MSN Messenger Service 1.0 clients and servers. Among the core services that the MSN Messenger Servers provide to clients are:

- Authenticated user logon.
- Adding and deleting members of the user's contact list.
- Changing the user's on-line state.
- Receipt of asynchronous, real-time, on-line state change notifications from members of the user's contact list.
- Delivering lightweight, real-time messages to other users.
- Receipt of asynchronous, real-time messages from other users.
- Configuring the user's access permissions, to restrict the ability of other users to view the user's on-line state or send messages to the user.

Additional background:

1. Some features extraneous to core instant messaging functionality contained within the MSN Messenger Service 1.0 protocol are beyond the scope of this document. Examples include client version management and directory functionality.

2. The purpose of this document is to provide the members of the IMPP working group with a reference implementation of a "monolithic" IM system. That is, a system designed for massive scale, but not yet capable of communication with servers other than those associated with this specific service. Since any standard in this area will of necessity be a "distributed" design that explicitly enables server-to-server and service-to-service communication, this document will serve primarily as a reference and example of one implementer's choices when providing IM functionality at scale.

3. This document reflects the protocol used in the 1.0 release of MSN Messenger clients and servers, deployed on the Internet in July of 1999. However, the service is in production and rapidly growing, which almost certainly will necessitate changes to the protocol as Microsoft gains operational experience with the service and expands its feature set. This Internet Draft may not be updated with such changes, and the changes may be made with little or no notice.

4. MSN Messenger Server Component Overview

MSN Messenger Service clients make connections to several different kinds of servers. They are separate components to facilitate running at scale - each component can be duplicated an arbitrary number of times, independently of each other, to enable large numbers of users.

4.1 Dispatch Server (DS)

The Dispatch Server is the initial point of connection between client and server. Its primary functions are protocol version negotiation, determination of which Notification Server (NS) is associated with the client making a connection (via an algorithm of the server's choosing), and referring the client to the proper NS.

4.2 Notification Server (NS)

The Notification Server is the primary server component. The client and the Notification Server authenticate, synchronize user properties, and exchange asynchronous event notifications. The client's connection to the Notification Server occurs after the referral from the Dispatch Server is completed, and persists without interruption during the user's MSN Messenger Service session.

Some of the events transmitted between a client and a Notification

Server are: State changes (e.g. client is on-line, client is offline, client is idle), Switchboard Server invitation requests (see below), and application-specific notifications that are beyond the scope of this document. (E.g. new e-mail has arrived)

4.3 Switchboard Server (SS)

The Switchboard Server is the component through which clients can establish lightweight communication sessions without requiring a direct network connection between clients. The common usage of the Switchboard Server is to provide instant messaging sessions. When a client wishes to communicate with another client, it sends a message to its Notification Server, which then refers the client to a Switchboard Server. Once the SS connection is established, the "destination" client receives a notification from its NS to connect to the same SS.

5. Protocol Conventions

5.1 Connection Type

The MSN Messenger Protocol currently works over TCP/IP. The MSN Messenger server components support connections over port numbers 1863, which is the registered port number assigned by the IANA (http://www.isi.edu/in-notes/iana/assignments/port-numbers).

5.2 Command Syntax

MSN Messenger Protocol command syntax is ASCII and single line-based. Commands begin with a case-sensitive, three-letter command type, followed by zero or more parameters, and terminated by CRLF. Parameters are separated by one or more whitespace characters and cannot contain whitespace characters. Parameters that contain spaces or extended (non 7-bit ASCII) characters should be encoded using URL-style encoding (e.g. "%20" for space). Some commands accept un-encoded binary data. In these cases, the length of the data is transmitted as part of the command, and the data is transmitted immediately following a CRLF of the command.

5.3 Asynchronous Requests

Commands issued from the client to the server that result in a reply are known as requests. Requests are entirely asynchronous. The client can submit several requests in sequence without waiting for the server response after submitting each request. The server is required to deliver a response or an error for each request

received, but it is not required to deliver the responses in the same order as the requests were received. The client can determine the request associated with a particular response by examining the Transaction ID parameter (described below).

5.4 User Handles

MSN Messenger Protocol uses User Handles for identifying users. A user handle (also known as "account name" and "logon name") is a text representation of the user's identity that is both unique and persistent. The user handle is syntactically equivalent to an e-mail address, and as such is subject to the same restrictions for character set, as described in RFC-822. Most notable among these restrictions are the limitation to Latin alphanumeric characters and a few symbols. The maximum acceptable length of the user handle is 129 bytes.

Implementation note: In the initial release of the client and server, user handles are Hotmail account names. All user handles must contain the "@hotmail.com" domain name, and user handles that do not contain a domain name are not valid.

5.5 Custom User Names

A custom user name (also known as "custom name" and "friendly name") is a user's representation of the "friendly" textual name associated with a user handle. (E.g. "Auntie Em" instead of em123@hotmail.com). Custom user names are neither unique nor persistent, and can contain any valid Unicode characters. Custom user names are represented in UTF-8 as described in RFC-2044 and URL-encoded as described in RFC-1738 when transmitted between the client and server. The maximum acceptable length of the encoded custom user name is 387 in the current implementation.

5.6 Transaction Identifiers

The Transaction Identifier (a.k.a. Transaction ID) is a numeric string representing a number between 0 and (2^32 - 1). It is a value that a client includes with any command that it issues to the server. In the current version of the protocol, the transaction identifier is used to associate server responses with client-issued commands. The server treats the transaction ID as an opaque number and does not assume any relationship between successive Transaction

IDs or any particular starting Transaction ID. It is the client's responsibility to guarantee the uniqueness of the Transaction IDs

for the purpose of disambiguating the commands and/or responses. (A future version of the protocol could enable the client to track the status or cancel a particular transaction using the transaction ID.)

When the server sends the response to a command to the client, it must include in the response the transaction ID that the client sent to the server when the client originally issued the command. In cases where a server sends a command to a client that requires a transaction ID but is not in response to a specific client command, it will use 0 as the transaction ID. In cases where a server sends multiple responses to a single client request, the server will use the same transaction ID in each response.

5.7 User List Types

Some of the protocol commands are used to the manipulate lists of users. The following types of user lists are supported by the protocol:

Forward List (FL) - The list of users for whom a given user wants to receive state change notifications. The Forward List is what is most commonly referred to as the user's "contact list."

Reverse List (RL) - The list of users who have registered an interest in seeing this user's state change notifications.

Allow List (AL) - The list of users who the user has explicitly allowed to see state change notifications and establish client-to-client sessions via a Switchboard Server.

Block List (BL) - The list of users who the user has explicitly prevented from seeing state change notifications and establishing client-to-client sessions via a Switchboard Server.

6. Command Summary Table

Command	From	To	Description
ACK	Switchboard	Client	Sends a positive message delivery acknowledgement.
ADD	Client Notification	Notification Client	Adds to the user's FL, AL, and BL. Notifies the client

			of asynchronous additions to a user's list.
ANS	Client	Switchboard	Accepts a request for a switchboard server session.
BLP	Client Notification	Notification Client	Changes the user's message privacy setting, which determines how to treat messages from users not already in the BL or AL.
BYE	Switchboard	Client	Notifies a client that a user is no longer in the session.
CAL	Client	Switchboard	Initiates a switchboard server session.
CHG	Client Notification	Notification Client	Sends a client state change to the server. Echoes the success of client's state change request.
FLN	Notification	Client	Notifies the client when users in the FL go off-line.
GTC	Client Notification	Notification Client	Changes the user's prompt setting, which determines how the client reacts to certain RL changes.
INF	Client Dispatch, Notification	Dispatch, Notification Client	Requests set of support authentication protocol from the server. Provides the set of supported authentication protocols to the client.
ILN	Notification	Client	Notifies the client of the initial online state of a user in the FL, while either logging on or adding

			a user to the FL.
IRO	Switchboard	Client	Provides the initial roster information for new users joining the session.
JOI	Switchboard	Client	Notifies a client that a user is now in the session.
LST	Client Notification	Notification Client	Retrieves the server's version of the user's FL, RL, AL, or BL.
MSG	Client	Switchboard	Sends a message to the members of the current session.
MSG	Notification, Switchboard	Client	Delivers a message from another client or from a server-side component.
NAK	Switchboard	Client	Sends a negative message delivery acknowledgement.
NLN	Notification	Client	Notifies the client when users in the FL go on-line or when their on-line state changes.
OUT	All	All	Ends a client-server Session.
REM	Client Notification	Notification Client	Removes from the user's FL, AL, and BL. Notifies the client of asynchronous removals from a user's list.
RNG	Notification	Client	Notifies the client of a request by another client to establish a session via a switchboard server.
SYN	Client Notification	Notification Client	Initiates client-server property synchronization.

USR	All	All	Authenticates client with server, possibly in multiple passes.
VER	Client Dispatch	Dispatch Client	Negotiates common protocol dialect between client and Server.
XFR	Client Notification	Notification Client	Requests a Switchboard server for use in establishing a session.
XFR	Dispatch Notification	Client Client	Notification of login-NS to the client or notification to move to a different NS.

7. Presence and State Protocol Details

This is a detailed list of protocol commands associated with presence functionality. They are defined in the order used by clients. Commands associated with instant messages are discussed in section 8 below.

7.1 Protocol Versioning

After the client connects to a dispatch server by opening a TCP socket to port 1863, the client and server agree on a particular protocol version before they proceed. The Client-Server protocol version handshake involves the following command exchange:

```
C: VER TrID dialect-name{ dialect-name...}
S: VER TrID dialect-name
```

The client can provide multiple dialect names in preferred order. The dialect-name parameter returned by the server is the version server is designating for this connection

The current protocol dialect-name supported by Messenger servers is "MSNP2". The dialect names are not case-sensitive.

The string "0" is a reserved dialect name and is used to indicate a failure response. E.g.:

```
S: VER TrID 0{ dialect-name ... }
```

7.2 Server Policy Information

The client next queries the server for variable "policy"

information. In this version of the protocol, the only policy

information returned by the server is the authentication package in use.

```
C: INF TrID
S: INF TrID SP{,SP...}
```

SP identifies a security package - the name of the SASL mechanism to use for authentication. "MD5" is used by the Notification Server, "CKI" by the Switchboard Server.

7.3 Authentication

The client needs to authenticate itself after protocol version handshake and identifying the security packages supported on the server. The following are the client server interactions involved.

```
C: USR TrID SP I{ AuthInitiateInfo}
S: USR TrID SP S{ AuthChallengeInfo}
C: USR TrID SP S{ AuthResponseInfo }
S: USR TrID OK UserHandle FriendlyName
```

The SP parameter is the name of the security package("MD5"). The next parameter is a sequence value, which must be I to (I)nitiate the authentication process and S for all (S)ubsequent messages. If authentication fails on the server, the client can start the authentication process again.

For the MD5 security package:

- The AuthInitiateInfo parameter provided by the client must be the User handle.
- The AuthChallengeInfo parameter returned by the server contains a challenge string.
- The AuthResponseInfo contains the binary response as a hexadecimal string, which the MD5 hash of the challenge and the User password strings concatenated together.

The final response from the server contains, in addition to the user handle, the current "Friendly Name" associated with the user handle. This is a "Custom User Name" as described above.

7.4 Referral

There are three cases in which clients are referred from one server to another:

1. The initial "Dispatch Server" refers the client to the Notification Server to which it is assigned.
2. Asynchronous referral by the Notification Server to reassign the client to a different Notification Server if that server is overloaded or undergoing maintenance.
3. During Switchboard Session establishment, the assigned Notification Server refers the client to a particular switchboard server for use. This is discussed below.

In the current implementation the Dispatch Server uses the user handle provided in the initial USR command above to assign the user in question to a Notification Server. Alternate implementations might not require referral at this stage.

If received, referral is of the form:

S: XFR TrID ReferralType Address[:PortNo]

ReferralType is either "NS" or "SB" and defines the type of referral to a Notification Server or Switchboard Server.
Address is a valid DNS name or IP address to a referred server, with optional port# suffixed as ":PortNo".

If this command is received from the server, the client should attempt to log in to the server provided.

In the case of "NS" referrals during logon, the Server automatically closes the client connection after sending this XFR response so that the client can connect to the new IP Address.

If sent asynchronously, the client is responsible for closing the connection.

After a "NS" referral, the client will not receive any more messages from the "old" NS, and also must not send any commands to the "old" NS after receiving an XFR.

7.5 Client User Property Synchronization

Several of the user properties used by the Messenger application are stored on the server. This is done for two reasons:

1) So that users can "roam", i.e. log in from different locations and still have the appropriate data, such as their contact lists and privacy settings.

2) If changes occur to a user's Reverse List while that user was offline (the user was added to another user's list), the client can be updated with this information.

For performance reasons it is useful to cache these properties on the client, so that bandwidth usage is minimized in the typical case where the user is not roaming and there were no Reverse List changes.

These requirements are met by the SYN command - synchronization.

Once a client logs in successfully, it uses the SYN command to ensure it has the latest version of the server-stored properties. These properties include: Forward List, Reverse List, Block List, Allow List, GTC setting (privacy setting when someone adds this user to their Forward List), and BLP setting (the user's privacy mode).

The SYN command is:

```
C: SYN TrID Ser#
S: SYN TrID Ser#
```

The Ser# parameter sent by the client is the version of the properties currently cached on the client. The server responds with the current server version of the properties. If the server has a newer version, the server will immediately follow the SYN reply by updating the client with the latest version of the user properties. These updates are done as described below, and are done without the client explicitly initiating a LST, GTC or BLP command. Note that the server will update all server-stored properties to the client, regardless of how many entries have been changed.

The following "List Retrieval and Property Management" section describes the format of the user properties sent by the server. After the SYN reply from the server, the user property updates will be sent from the server in this sequence: GTC, BLP, LST FL, LST AL, LST BL, LST RL.

All the user property updates will share the same TrID as the SYN command and reply.

7.6 List Retrieval And Property Management

Synchronizing can result in a batch of user properties and lists getting sent by the server to the client. However, the client application can also initiate a request to retrieve the server-

stored lists and properties. The following are the privacy property and list retrieval commands. The response formats are the same whether it is a client-initiated request, or whether it is a response to the SYN process as described above.

List Command

By issuing the LST command, the client can explicitly request that a list be sent. The server will respond with a series of LST responses, one LST response for each item in the requested list.

```
C: LST TrID LIST
S: LST TrID LIST Ser# Item# TtlItems UserHandle CustomUserName
```

- LIST is FL/RL/AL/BL for Forward List, Reverse List, Allow List, and Block List, respectively.
- The Item# parameter contains the index of the item described in this command message. (E.g. item 1 of N, 2 of N, etc.)
- The TtlItems parameter contains the total number of items in this list.
- UserHandle is the user handle for this list item.
- CustomUserName is the friendly name for this list item.

If the list is empty, the response will be:

```
S: LST TrID LIST Ser# 0 0
```

Reverse List Prompting

The client can change its persistent setting for when to prompt the user in reaction to an Reverse List change. This is accomplished via the GTC command:

```
C: GTC TrID [A | N]
S: GTC TrID Ser# [A | N]
```

The value of the A/N parameter determines how the client should behave when it discovers that a user is in its RL, but is not in its AL or BL. (Note that this occurs when a user has been added to another user's list, but has not been explicitly allowed or blocked):

A - Prompt the user as to whether the new user in the RL should be added to the AL or the BL
N - Automatically add the new user in the RL to the AL

The A/N parameter is not interpreted by the server, merely stored.

The server will respond with the current setting if the change was successful. Otherwise, it will return an error with the matching TrID. If the client tries to change the setting to the same value as the current setting, the server will respond with an error message.

The default setting is A when a new user connects to the server for the first time.

Privacy Mode

The client can change how the server handles instant messages from users via the BLP command:

```
C: BLP TrID [AL | BL]
S: BLP TrID Ser# [AL | BL]
```

The AL/BL parameter determines how the server should treat messages (MSG and RNG) from users. If the current setting is AL, messages from users who are not in BL will be delivered. If the current setting is BL, only messages from people who are in the AL will be delivered.

The server will respond with the current setting if the change was successful. Otherwise, it will return an error with the matching TrID. If the client tries to change the setting to the same value as the current setting, the server will respond with an error message.

The default setting is AL when a new user connects to the server for the first time.

7.7 Client States

After the client is authenticated and synchronized, the client establishes its initial state with the server with the CHG command. The syntax of the command is:

```
C: CHG TrID State
S: CHG TrID State
```

When the state is changed, the server will echo the settings back to client. The state shall not be considered changed until the response

is received from the server.

Note that the server can send a state change message to the client at any time. If the server changes the state without a request from the client, the TrID parameter will be 0.

States are denoted by a string of three characters. The predefined states that the server recognizes are:

NLN - Make the client Online (after logging in) and send and receive notifications about buddies.
FLN - Make the client Offline. If the client is already online, offline notifications will be sent to users on the RL. No message activity is allowed. In this state, the client can only synchronize the lists as described above.
HDN - Make the client Hidden/Invisible. If the client is already online, offline notifications will be sent to users on the RL. The client will appear as Offline to others but can receive online/offline notifications from other users, and can also synchronize the lists. Clients cannot receive any instant messages in this state.

All other States are treated as sub-states of NLN (online). The other States currently supported are:
BSY - Busy.
IDL - Idle.
BRB - Be Right Back.
AWY - Away From Computer.
PHN - On The Phone.
LUN - Out To Lunch.

7.8 List Modifications

The protocol supports generic commands to add and remove users from various lists. This is used by clients to enable "Adding" contacts to the list of folks being watched, or for the "Block" and "Allow" features that define how users chooses to interact with one another.

However, these generic commands have different semantics based on the list being modified. For example, only the server can add or remove entries from the Reverse List - since it is an indirect consequence of the user having been added to another user's Forward List.

The add and remove commands:

```
C: ADD TrID LIST UserHandle CustomUserName
S: ADD TrID LIST ser# UserHandle CustomUserName

C: REM TrID LIST UserHandle
S: REM TrID LIST ser# UserHandle
```

Valid values for LIST in Client initiated adds and removes are FL/AL/BL.

All client initiated adds and removes will be echoed by the server with a new serial number that should be persisted by the client along with the list modification. If not successful, an error will result.

The protocol also supports the concept of an ADD or REM that the client did not initiate. Server generated ADDs and REMs can have LIST values of FL/AL/BL/RL. This is common with RL changes, which are never initiated by the client, but is an indirect consequence of this user having been added to someone's Forward List. If the RL change happens while the user is online, it will trigger an asynchronous ADD or REM command from the server.

Asynchronous ADDs and REMs to the FL, AL, and BL can happen when the server allows an authenticated user to make list changes from another environment, such as a web site. In all of these cases, the server will send the ADD or REM command with the TrID parameter equal to 0.

7.9 Notification Messages

The client receives asynchronous notifications whenever a contact on the user's Forward List changes its state. The notifications are of the form:

```
S: NLN Substate UserHandle FriendlyName
S: ILN TrID Substate UserHandle FriendlyName
S: FLN UserHandle
```

NLN indicates that a user has come online.
- Substate can be any three-letter code (see "Client States" above).
- UserHandle and FriendlyName are the handle and names associated with the user coming online.

ILN is similar to the NLN message, and is received from the server in response to an CHG or ADD command from the client:

1. Immediately after the client logon and sends its first CHG command to the NS. In this case several ILNs may be received - one for each Forward List contact that is currently online.
2. After the client sends an "ADD TrID FL UserHandle CustomUserName" to the NS. (e.g. ILN for the new contact if that contact is currently online)

In both cases, TrID in the ILN is the same as the one sent by the client in the CHG or ADD command.

FLN means that the specified user is now offline.

7.10 Connection Close

The client issues the following command to logoff from the NS:

```
C: OUT
S: OUT {StatusCode}
```

The server will reply with an OUT to the client before it initiates a disconnect, with an optional StatusCode.

The StatusCode can be "OTH", which indicates that a client with the same user handle and password has logged on to the server from another location, or "SSD" meaning the server is being shut down for maintenance.

The server will drop the connection after sending the OUT.

7.11 Error Information

Error messages from the server are of the format:

```
S: eee {TrID} {(error-info) {param...}}
```

eee is a 3 digit decimal number indicating the error code. Error-info contains the description of the error in a text string localized to the server's locale. The optional parameters provide indication of the client command causing the error. TrID is the Transaction ID of the client command that caused this error. Any server generated errors will not have Transaction IDs.

ERR_SYNTAX_ERROR	200
ERR_INVALID_PARAMETER	201
ERR_INVALID_USER	205
ERR_FQDN_MISSING	206
ERR_ALREADY_LOGIN	207
ERR_INVALID_USERNAME	208
ERR_INVALID_FRIENDLY_NAME	209
ERR_LIST_FULL	210
ERR_ALREADY_THERE	215

ERR_NOT_ON_LIST	216
ERR_ALREADY_IN_THE_MODE	218
ERR_ALREADY_IN_OPPOSITE_LIST	219
ERR_SWITCHBOARD_FAILED	280
ERR_NOTIFY_XFR_FAILED	281
ERR_REQUIRED_FIELDS_MISSING	300
ERR_NOT_LOGGED_IN	302
ERR_INTERNAL_SERVER	500
ERR_DB_SERVER	501
ERR_FILE_OPERATION	510
ERR_MEMORY_ALLOC	520
ERR_SERVER_BUSY	600
ERR_SERVER_UNAVAILABLE	601
ERR_PEER_NS_DOWN	602
ERR_DB_CONNECT	603
ERR_SERVER_GOING_DOWN	604
ERR_CREATE_CONNECTION	707
ERR_BLOCKING_WRITE	711
ERR_SESSION_OVERLOAD	712
ERR_USER_TOO_ACTIVE	713
ERR_TOO_MANY_SESSIONS	714
ERR_NOT_EXPECTED	715
ERR_BAD_FRIEND_FILE	717
ERR_AUTHENTICATION_FAILED	911
ERR_NOT_ALLOWED_WHEN_OFFLINE	913
ERR_NOT_ACCEPTING_NEW_USERS	920

8. Session based Instant Messaging Protocol Details

MSN Messenger Service utilizes a lightweight, session-based messaging scheme. In order for two clients to exchange instant messages, they must first establish a common session via a Switchboard Server. They can invite additional clients to join the established session.

8.1 Referral to Switchboard

This process begins with a "calling" client requesting a referral from its Notification Server to a Switchboard Server:

```
C: XFR TrID SB
S: XFR TrID SB Address SP AuthChallengeInfo
```

- SB is the type of referral being requested or granted.
- Address is the DNS name or IP address of a Switchboard Server that has been assigned, and that the client should connect to.
- SP is the Security Package being used. In this version of the protocol it is "CKI" only.
- AuthChallengeInfo is a cookie that the client needs to present to the Switchboard server for authentication.

8.2 Switchboard Connections and Authentication

After the XFR reply is received, the client makes a TCP/IP connection to the Switchboard server using port 1863. Note that a lack of version negotiation in the switchboard connection is a limitation of the current implementation.

The client first needs to authenticates with the Switchboard Server:

```
C: USR TrID UserHandle AuthResponseInfo
S: USR TrID OK UserHandle FriendlyName
```

- AuthResponseInfo is the cookie for CKI security package returned by the Notification Server in the XFR.
- UserHandle and FriendlyName are the Switchboard's echoes of the user handle and friendly name of the user.

8.3 Inviting Users to a Switchboard Session

Any user in a Switchboard session can invite other users to join the session. The CAL command is sent to the Switchboard server for this purpose:

```
C: CAL TrID UserHandle
S: CAL TrID Status SessionID
```

The Messenger servers verify that the calling user has permissions to contact the called user, with consideration given to the called user's privacy settings and its online state. If instant messaging with this user is not allowed, the server responds to the calling user with an error. If it is allowed, the Switchboard server causes a RNG command to be sent to the called client (see below), and returns a CAL echo to the calling client. The CAL echo has these parameters:

- Status is a predefined status code - in this implementation it must be "RINGING".
- SessionID is the ASCII representation of a decimal number that

uniquely identifies this session on the Switchboard Server.

8.4 Getting Invited to a Switchboard Session

The other side of the session establishment is the behavior of the called client. The called client receives a RNG from its Notification Server and is expected to connect to the Switchboard Server and respond with an ANS.

The client receives a RNG from the Notification Server as follows:

```
S: RNG SessionID SwitchboardServerAddress SP AuthChallengeInfo
   CallingUserHandle CallingUserFriendlyName
```

- SessionID is a numeric ASCII session ID.

- SwitchboardServerAddress is a DNS name or IP Address
- SP is the security package in use. In this implementation only "CKI" is supported.
- AuthChallengeInfo is the cookie to be passed back to the switchboard to gain entrance to the session.
- CallingUserHandle is the user handle of the caller.
- CallingUserFriendlyName is the custom user name of the caller.

To join the session, the called client connects to the Switchboard Server and carries out the following exchange to join the session:

```
C: ANS TrID LocalUserHandle AuthResponseInfo SessionID
S: IRO TrID Participant# TotalParticipants UserHandle
   FriendlyName
S: ANS TrID OK
```

The IRO commands relay to the newly joined client roster information about the current session. Each IRO command message from the Switchboard contains one participant in the session.

- Participant# contains the index of the participant described in this IRO command (e.g. 1 of N, 2 of N).
- TotalParticipants contains the total number of participants currently in the session.

The entire session roster will be sent to the new client joining the session before any JOI or BYE commands described below.

If no one is in the session when the user joins (an unexpected error condition), the server skips directly to "ANS TrID OK" command. All the responses from the server related to the issued ANS command will contain the same TrID as the original client ANS request.

8.5 Session Participant Changes

When a new user joins a Switchboard session, the server sends the following command to all participating clients, including the client joining the session:

S: JOI CalleeUserHandle CalleeUserFriendlyName

- CalleeUserHandle is the user handle of the new participant.
- CalleeUserFriendlyName is the Custom User Name of the new participant.

If a client's connection with the Switchboard Server is dropped for any reason, the server sends the following command to the remaining clients in the session:

S: BYE CalleeUserHandle

- CalleeUserHandle is the user handle of the participant that left the session.

Privacy Note:
If the client moved a contact to the BL while Switchboard sessions are active, it is the client's responsibility to leave any session that should now be blocked. The servers only enforce privacy permissions when inviting users to a session. Further, the servers only enforce privacy permission with respect to the calling user, and not the other participants in a Switchboard session. Therefore, in a multipoint session, it is possible for a user to participate in a session with someone whom he has blocked, if a third party is performing the invitation.

8.6 Leaving a Switchboard Session

When a client wishes to disconnect from the session, it sends the following command and waits for the Switchboard to close the connection:

C: OUT

8.7 Instant Messages

Sending an Instant Message

Once a client-to-client session has been established via the

Switchboard Server, sending an Instant Message to the participants of the session is done as follows:

```
C: MSG TrID [U | N | A] Length\r\nMessage
S: NAK TrID
S: ACK TrID
```

U, N, and A correspond to the three delivery acknowledgement modes: Unacknowledged, Negative-Acknowledgement-Only, and Acknowledgement. Depending on the value of this parameter, either nothing, NAK, or ACK will be sent back by the Switchboard Server to the client.

For Unacknowledged mode, the Switchboard Server does not respond to the sending client with the success or failure of message delivery.

For Negative-Acknowledgement-Only mode, the Switchboard Server responds to the send client only if the message could not be delivered to the recipient client.

Acknowledgement mode is not currently implemented.

Length is the length of the Message parameter in bytes, whereas Message is the actual message as described below.

8.8 Receiving an Instant Message

A client can receive a system-generated message from the Notification Server, or it can receive an instant message from

another client via a Switchboard Server. The message is received in the following format:

```
S: MSG UserHandle FriendlyName Length\r\nMessage
```

The UserHandle and FriendlyName are those of the sending user. Length is the length of the message in bytes.

Message is a MIME encoded stream, using a standard MIME header as defined by RFC-1521 and RFC-822.

Message is constructed as:

```
MIME-Header\r\nMIME-Header\r\n\r\nMessageData
```

MIME-Header is constructed as:

```
     string": "string
     (E.g. "Content-Type: text/plain")
```

The Content-Type MIME headers that the current client will use and recognize are:

```
     "text/plain;charset=UTF-8"
     "text/plain"
```

If "charset=UTF-8" appears at the end of the Content-Type, the Message Data is UTF-8 encoded.

Note: The Switchboard Server does not interpret the contents of the Message.

9. Author's Addresses

Ramu Movva
Microsoft Corporation
One Microsoft Way
Redmond WA 98052
ramum@microsoft.com

William Lai
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
wlai@microsoft.com