



Skype API

Version 1.4



The whole world can talk for free. [Skype.com](https://www.skype.com)

About this guide

This section describes this guide to help you find what you want quickly. If you are developing for the Skype API, please read the complete document and understand its technical and procedural information.

Purpose of this guide

This document describes the public Skype application programming interface (API), version 1.4 for Windows, the Skype APIs for Linux and Mac, and provides a reference guide for the Skype developer community.

Who reads this guide?

Skype's developer community who work with us to enrich the Skype experience and extend the reach of free telephone calls on the internet.

What is in this guide?

This document contains the following information:

- [Overview of the Skype API](#)
- [Using the Skype API on Windows](#)
- [Using the Skype API on Linux](#)
- [Using the Skype API on Mac](#)
- [Skype protocol](#)
- [Skype reference](#)
 - [Terminology](#)
 - [Commands](#)
 - [Objects](#)
 - [Object properties](#)
 - [General parameters](#)
 - [Notifications](#)
 - [Error codes](#)
- [Skype URI](#)
- [Skype release notes](#)
- [Skype commands list](#)

More information

The Skype websites contain useful information for developers:

- The [Skype Developer Zone](#) is where to find all the Skype public API docs (including earlier versions) and where you can download free examples for Windows, Linux, and Mac platforms.
- Share ideas and information on the [Skype Public API forum](#)
- See the cool things people are creating and submit your project to the [Skype Extras Gallery](#).

Legal information

This document is the property of Skype Technologies S.A. and its affiliated companies (Skype) and is protected by copyright and other intellectual property rights laws in Luxembourg and abroad. Skype makes no representation or warranty as to the accuracy, completeness, condition, suitability, or performance of the document or related documents or their content, and shall have no liability whatsoever to any party resulting from the use of any of such documents. By using this document and any related documents, the recipient acknowledges Skype's intellectual property rights thereto and agrees to the terms above, and shall be liable to Skype for any breach thereof. For usage restrictions please read the [end user license agreement](#) (EULA).

Text notation

This document uses `monospace` font to represent code, file names, commands, objects and parameters. The following text conventions apply for syntax:

- `CALL` - uppercase text indicates a keyword, such as command, notification, and object.
- `property` - lowercase text indicates a category of a keyword
- `<username>` - angle brackets indicate an identifier, such as username or call id
- `[<target>]` - square brackets identify optional items
- `*` - asterisk indicates repetitive items
- `|` - vertical bar means "or"
- `->` - command issued by client (used in examples)
- `<-` - response or notification from Skype (used in examples)
- `//` - comment line (used in examples)

Overview of the Skype API

The Skype API passes commands in simple text messages between Skype and client applications and devices. Client applications can be applications which extend Skype functionality and devices can be hardware or software devices, such as USB phones. The Skype API has two main components; the Skype phone API and the Skype access API.

Skype protocol

The Skype protocol defines communications in Skype. Skype releases new versions of the protocol periodically to enable new communication features in Skype. Skype always uses the correct protocol version to match the protocol version reported by a connecting device or application. Skype never responds with a higher protocol than the connecting device or application and the default protocol it uses is version 1.

Skype phone API

The Skype phone API provides an interface to connecting devices such as USB phones. The Skype client controls the API and triggers events on the device (or application) driver. The connecting device receives abstract events such as:

- the green button is pressed
- the handset is off-hook
- the device rings

During installation, Skype detects compatible audio drivers and installs them. If a new device and driver are added after Skype installation, Skype detects the new driver but the user must select the driver manually for it to become operational.

The Skype phone API uses the following commands to communicate with Skype, substituting values and setting indicators as appropriate:

- NAME `deviceName`
- PROTOCOL `version`
- AUDIO_IN `deviceName`
- AUDIO_OUT `deviceName`
- HOOK ON|OFF
- MUTE ON|OFF
- BTN_PRESSED (0-9, A-Z, #, *, +, UP, DOWN, YES, NO, SKYPE)
- BTN_RELEASED ...

The Skype phone API uses the `MUTE ON|OFF` command to control the mute function on a device.

Skype access API

The Skype access API enables external applications to control certain Skype functions, for example to place a call or to get a Skype user profile. In the interests of privacy and security, before an external application can take control, Skype pops up the name of the application to the user and asks if it is OK to allow access.

When a client application starts controlling Skype, Skype switches audio devices to the devices reported by the controlling client.

The Skype access API has the following characteristics:

- All actions performed using the API are mirrored on the Skype client.
- Multiple applications can use the Skype access API at the same time.
- All times and dates in the API are in UTC (Coordinated Universal Time)
- Transmission over the API is in UTF-8 encoding.

Using the Skype API on Windows

When developing applications to work with Skype, follow these general guidelines:

- Give intuitive names to executable files (.exe files) because this name is displayed to the user for confirmation. If the name is unclear, the user might not allow the application to access Skype.
- Sign applications with VeriSign's CodeSigning certificate.
- The application must support the `NAME` command and publish its name

Skype for Windows sends and receives API commands using `WM_COPYDATA` messages. Use the `RegisterWindowMessage` method to register the following messages:

- `SkypeControlAPIDiscover`
- `SkypeControlAPIAttach`

To initiate communication, a client application broadcasts the `SkypeControlAPIDiscover` message, including its window name as a `wParam` parameter. Skype responds with a `SkypeControlAPIAttach` message to the specified window name and indicates the connection status with one of the following values:

- `SKYPECONTROLAPI_ATTACH_SUCCESS = 0` - The client is attached and the API window handle is provided in `wParam` parameter.
- `SKYPECONTROLAPI_ATTACH_PENDING_AUTHORIZATION = 1` - Skype acknowledges the connection request and is waiting for user confirmation. The client is not yet attached and must wait for the `SKYPECONTROLAPI_ATTACH_SUCCESS` message.
- `SKYPECONTROLAPI_ATTACH_REFUSED = 2` - The user has explicitly denied access to client.
- `SKYPECONTROLAPI_ATTACH_NOT_AVAILABLE = 3` - The API is not available at the moment, for example because no user is currently logged in. The client must wait for a `SKYPECONTROLAPI_ATTACH_API_AVAILABLE` broadcast before attempting to connect again.

When the API becomes available, Skype broadcasts the `SKYPECONTROLAPI_ATTACH_API_AVAILABLE = 0x8001` message to all application windows in the system. The data exchange uses commands (or responses), provided as null-terminated UTF-8 strings. The terminating 0 must be transferred as well. You cannot combine several messages in one packet. There is no limit to the length of the transferred string.

Note: The result of processing the message must be different from zero (0), otherwise Skype considers that the connection broken.

If the API client spends more than 1 second processing a message, the connection is disconnected. Use the `PING` command to test the connection status. To ease debugging during development, in regedit enter the key `APITimeoutDisabled` (DWORD value, 0 = timeout enabled 1 = timeout disabled) into the `HKCU\Software\Skype\Phone\UI` file in the registry to override the 1 second timeout.

To check if Skype is installed, in regedit check if the following key exists: `HKCU\Software\Skype\Phone, SkypePath`. This key points to the location of the `skype.exe` file. If this key does not exist, check if the `HKLM\Software\Skype\Phone, SkypePath` key exists. If the `HKCU` key does not exist but the `HKLM` key is present, Skype has been installed from an administrator account but not been used from the current account.

Download free examples

- [Example code for C++](#)

Using the Skype API on Linux

The Skype API for Linux uses the Skype protocol, version 1.1.0.3. The API uses the d-bus messaging framework and you can download the d-bus libraries from freedesktop.org.

If RPM Package Manager is installed on the Linux device, the d-bus files are automatically configured. If RPM is not installed, create a text file named `skype.conf`, and save it to `/etc/dbus-1/system.d/skype.conf`.

The configuration file enables the Skype API to use the d-bus framework and contains the following:

```
<!DOCTYPE busconfig PUBLIC "-//freedesktop//DTD D-BUS Bus
Configuration 1.0//EN"
"http://www.freedesktop.org/standards/dbus/1.0/busconfig.dtd">
<busconfig>
  <policy context="default">
    <allow own="com.Skype.API"/>
    <allow send_destination="com.Skype.API"/>
    <allow receive_sender="com.Skype.API"/>
    <allow send_path="/com/Skype"/>
  </policy>
</busconfig>
```

To access the Skype API from a client application:

- The application passes its name to Skype:

```
<- NAME clientApplication
```

- Skype pops up a confirmation window to the user with the following response:

```
ClientApplication wants to talk to Skype. OK?
```

•

- If OK, protocol messages are exchanged:

```
-> OK
<- PROTOCOL 1
-> PROTOCOL 1
```

The Skype protocol manages the subsequent session. The d-bus service is `com.Skype.API`

Two communication paths are used:

- for client-to-Skype commands the path is `/com/Skype`
- for Skype-to-client commands the path is `/com/Skype/Client`

Two methods are used:

- use the `Invoke` method with one string parameter for client-to-Skype commands
- use the `Notify` method for Skype-to-client commands and responses

More information and examples

Read the [Skype API for Linux guide](#) and download free examples for C and for Python.

Using the Skype API on Mac

The Skype API for Mac has interfaces for Cocoa, Carbon, and AppleScript. The Cocoa and Carbon interfaces are implemented in `Skype.framework`. Skype recommends that you include the Skype framework in your application as an embedded framework. To do so, copy it into the application bundle and link it to the application.

Client applications send string commands to control Skype. The format of these strings commands is described in the [Skype API reference](#). If you are using a Cocoa or Carbon interface, Skype will send information back to your application by calling asynchronous delegate functions/methods.

More information and examples:

Read the [Skype API for Mac](#) guide and download the Skype framework and free examples.

Skype protocol

The Skype protocol is currently in its fifth version. Starting with protocol 1, a new version is created only when new commands become incompatible with existing commands. The protocol number does not increase when commands are introduced but existing commands are unchanged.

Protocol 5

Protocol 5 is the current version of the Skype protocol and is used by the following versions of Skype:

- 1.4.0.84 - Windows
- 1.3.0.33 - Windows and Mac

This protocol introduced multiperson chat commands.



Protocol 4

Protocol 4 is used by the following versions of Skype:

- 1.2.0.11 - Windows
- 1.1.0.3 Windows and Linux

This protocol introduced ISO code prefixes for language and country.

Protocol 3

Protocol 3 is used by the following version of Skype:

- 1.1.0.61 - Windows

This protocol introduced a compatibility layer for previous instant messaging.

Protocol 2

Protocol 2 is used by the following version of Skype:

- 1.0.0.94

This protocol implemented the following changes:

- Introduced the `SKYPEME` online status
- For calls on hold, notifies clients with either `LOCALHOLD` or `REMOTEHOLD`. Protocol 1 simply returned `ONHOLD`.
- Introduces the call status, `CANCELLED`.

Protocol 1 and 2 compatibility

If the requested protocol is smaller than 3, all incoming commands are converted as follows:

- `SEARCH MESSAGES` -> `SEARCH CHATMESSAGES`
- `SEARCH MISSEDMESSAGES` -> `SEARCH MISSEDCCHATMESSAGES`
- `GET MESSAGE` -> `GET CHATMESSAGE`
- `SET MESSAGE` -> `SET CHATMESSAGE`

The `GET MESSAGE` properties are also converted:

- `PARTNER_HANDLE` -> `FROM_HANDLE`
- `PARTNER_DISPNAME` -> `FROM_DISPNAME`

All API notification (including `GET/SET MESSAGE`) replies are converted:

- `CHATMESSAGE * FROM_HANDLE x` -> `MESSAGE * PARTNER_HANDLE x`
- `CHATMESSAGE * FROM_DISPNAME x` -> `MESSAGE * FROM_DISPNAME x`
- `CHATMESSAGE * property x` -> `MESSAGE * property x`

If the protocol is less than 3, `SEARCH MESSAGES` and `SEARCH MISSEDMESSAGES` commands return string `MESSAGES 1, 2, 3`.

Skype API reference

The Skype API reference is a guide for developers working with the public Skype API. The reference contains the following information:

- [Terminology](#)
- [Commands](#)
- [Objects](#)
- [Object properties](#)
- [General parameters](#)
- [Notifications](#)
- [Error codes](#)

Terminology

The Skype API reference uses the following terms:

- The Skype access API is also known as the Skype control API.
- The client application issues a **command** to control Skype.
- In reply to some commands, Skype returns a synchronous **response**. Not all commands require a response. Responses are documented under their relevant commands.
- Skype **objects** are application, call, chat, chatmessage, profile, user, and voicemail.
- A **notification** is an asynchronous message Skype sends to a client when a change occurs, for example when a contact comes online or a new chatmessage is received.
- Skype has **general parameters** to control the setup, current user and connection information.
- **Connectable users** are online Skype users who are in the client contact list and also non-contacts who are in active communication with the client.

Commands

This section provides a reference to the commands used in Skype, and contains the following:

- [Description of command identifiers](#)
- [Commands for making and managing calls.](#)
- [Commands for leaving and manipulating voicemails.](#)
- [Commands for creating chats and sending messages](#)
- [Search commands](#)
- [Commands for managing history](#)
- [Commands for controlling the Skype user interface](#)
- [Application to application commands](#)
- [Call forwarding](#)
- [Testing connections](#)

Command identifiers

A command identifier is useful to identify a response to a specific command. A command identifier is supported by most commands and is included in the response.

Syntax

#<command_id> command

Response

#<command_id> response|error

Parameters

command_id - client assigned alphanumeric identifier

Errors

all possible errors for a given command

Version

Protocol 4

Notes

- A command identifier is not included in asynchronous notification events initiated by a command.
- Asynchronous commands usually return a synchronous response with the command id. When the command is processed an asynchronous notification is also sent
- A response may come not directly after the command because there can other messages can be received between command and response.

Examples

- Simple response to command

```
-> #AB GET USERSTATUS
<- #AB USERSTATUS ONLINE
```

- Invalid command with reported error

```
-> #123 GET XZY
<- #123 ERROR 7 GET: invalid WHAT
```

- Command response and notification

```
-> #cmd11 SET USERSTATUS ONLINE
// this is the response for the command
<- #cmd11 USERSTATUS ONLINE
// this is notification when the command is actually processed
<- USERSTATUS ONLINE
```

- Command response and notification are asynchronous

```
-> #50 CALL +18005551234
// note that events can arrive before response
<- CALL 651 STATUS ROUTING
<- #50 CALL 651 STATUS ROUTING
<- CALL 651 PSTN_STATUS 10503 Service Unavailable
// the following events do not have a command id
<- CALL 651 FAILUREREASON 1
<- CALL 651 STATUS FAILED
```

- Notifications can appear between command-response

```
-> #50 PING
// note that other events can arrive before command response
<- USER echo123 LASTONLINETIMESTAMP 1105764678
<- USER echo123 FULLNAME Echo Test Service
<- USER test LASTONLINETIMESTAMP 1105487965
// Now comes Skype response to command
```

<- #50 PONG

Making and handling calls

This section describes the commands for making and handling calls. The commands are:

- Use the [CALL command](#) to make a call.
- Use the [GET CALL command](#) to retrieve call information.
- Use the [SET CALL INPROGRESS command](#) to take an incoming call.
- Use the [SET CALL FINISHED command](#) to end a call.
- Use the [SET CALL ONHOLD command](#) to place a call on hold.
- Use the [SET CALL JOIN CONFERENCE command](#) to join a call to a conference.
- Use the [SET CALL DTMF command](#) to set DTMF for a call.
- Use the [SET CALL SEEN command](#) to mark a missed call as seen in order to clear the missed events list.

Refer to [call failure reasons](#) for a list of all reasons for call failure.

CALL

Syntax

```
CALL <target>[, <target>]*
```

Response

```
CALL <call_ID> <status>
```

Parameters

<target> - targets to be called. In case of multiple targets conference is created. Available target types:

- USERNAME - Skype username, e.g. "pamela", "echo123"
- PSTN - PSTN phone number, e.g. "+18005551234", "003725555555"
- SPEED DIAL CODE - 1 or 2 character speed-dial code

Errors

- ERROR 34 invalid user handle
Target username/number missing or contains invalid characters
- ERROR 39 user blocked
Trying to call to a blocked user (unblock user in contactlist)
- ERROR 73 too many participants
Call is initiated to more than 4 people
- ERROR 92 call error
Call is initiated to a number that is neither PSTN number nor a speedial number

Version

Protocol 1

Notes

The Skype call window is focused when a call is initiated through the API. It is possible to make speed dial calls via the API.

Example

```
-> CALL echo123
<- CALL 1402 STATUS ROUTING
<- CALL 1402 SUBJECT
<- CALL 1402 STATUS ROUTING
<- CALL 1402 STATUS RINGING
<- CALL 1402 STATUS INPROGRESS
<- CALL 1402 DURATION 1
<- CALL 1402 STATUS FINISHED
```

GET CALL

Syntax

```
GET CALL <id> property
```

Response

```
CALL <id> property <value>
```

Parameters and response values

refer to the [CALL object](#)

Errors

- ERROR 7 GET: invalid WHAT
Object name missing or misspelled
- ERROR 11 Invalid call id
ID includes characters other than numeric characters
- ERROR 12 Unknown call
Call with specified ID does not exist in current user's call history
- ERROR 13 Invalid PROP
Property name missing or misspelled
- ERROR 71 Invalid conference participant NO
Conference participant's number is not a number or is too long

Version

Protocol 1

SET CALL INPROGRESS

Syntax

```
SET CALL <id> STATUS INPROGRESS
```

Parameters

<id> - call ID (numeric)

Errors

- ERROR 19 Invalid call id
ID includes other than numeric characters
- ERROR 20 Unknown call
Call with specified ID does not exist
- ERROR 23 Cannot resume this call at the moment
Given call is not ringing and therefore can not be answered.

SET CALL FINISHED

Syntax

```
SET CALL <id> STATUS FINISHED
```

Parameters

<id> - call ID (numeric)

Errors

- ERROR 19 Invalid call id
ID includes other than numeric characters
- ERROR 20 Unknown call
Call with specified ID does not exist in current user's call history nor is active.
- ERROR 24 Cannot hangup inactive call
Given call is not in progress and therefore can not be hung up.

SET CALL ONHOLD

Syntax

```
SET CALL <id> STATUS value
```

Parameters

- <id> - call ID (numeric)
- value
 - ONHOLD - hold call
- INPROGRESS - resume call

Errors

- ERROR 19 Invalid call id
ID includes other than numeric characters
- ERROR 20 Unknown call
The call ID does not exist in current user's call history nor is it active.
- ERROR 22 Cannot hold this call at the moment
Given call is not in progress and therefore can not be placed on hold.
- ERROR 23 Cannot resume this call at the moment
Given call is not in hold and therefore can not be resumed.

SET CALL JOIN CONFERENCE

Syntax

```
SET CALL <joining_id> JOIN_CONFERENCE <master_id>
```

Response

```
CALL <id> CONF_ID <conference_id>
```

Parameters

- <joining_id> - call ID (numeric) to join into;
- <master_id> - master call ID, where is another call's ID.

Errors

- ERROR 19 Invalid call id
ID includes other than numeric characters
- ERROR 20 Unknown call
Call with specified ID does not exist in current user's call history nor is active.
- ERROR 72 Cannot create conference
Creating conference, for example "SET CALL 65 JOIN_CONFERENCE 66" fails for some reason.

Note

- It is possible to initiate a conference with the CALL target1, target2 command

Example

```
// make first call
-> CALL test
<- CALL 1540 STATUS ROUTING
<- CALL 1540 SUBJECT
<- CALL 1540 STATUS ROUTING
<- CALL 1540 STATUS RINGING
<- CALL 1540 STATUS INPROGRESS
// set first call on hold ...
-> SET CALL 1540 STATUS ONHOLD
<- CALL 1540 STATUS INPROGRESS
<- CALL 1540 STATUS ONHOLD
// .. and make another call
-> CALL echo123
<- CALL 1545 STATUS ROUTING
<- CALL 1545 SUBJECT
<- CALL 1545 STATUS ROUTING
<- CALL 1545 STATUS RINGING
<- CALL 1545 STATUS INPROGRESS
// join second call (1545) into conference with first call (1540)
-> SET CALL 1545 JOIN_CONFERENCE 1540
<- CALL 1545 CONF_ID 17930
<- CALL 1545 CONF_ID 17930
<- CALL 1540 CONF_ID 17930
// first call is automatically resumed and joined to conference
<- CALL 1540 STATUS INPROGRESS
// ...
<- CALL 1540 DURATION 53
<- CALL 1540 STATUS FINISHED
<- CALL 1545 DURATION 23
<- CALL 1545 STATUS FINISHED
```

SET CALL DTMF

Syntax

```
SET CALL <id> DTMF <value>
```

Parameters

- <id> - call ID (numeric)
- <value> - sends value as DTMF. Permitted symbols in VALUE are: {0..9,#,*}.

Notes

DTMF support and quality for PSTN calls depends on terminating partner.

Errors

- ERROR 19 Invalid call id
ID includes other than numeric characters
- ERROR 20 Unknown call
Call with specified ID does not exist in current user's call history nor is it active.
- ERROR 21 Unknown/disallowed call prop
DTMF property value is incorrect or misspelled

SET CALL SEEN

Syntax

```
SET CALL <id> SEEN
```

Response

```
CALL <id> SEEN TRUE
```

Parameters

<id> - call ID (numeric)

Errors

- ERROR 19 Invalid call id
ID includes other than numeric characters
- ERROR 20 Unknown call
Call with specified ID does not exist in current user's call history nor is active.

Example

```
-> SET CALL 15 SEEN  
<- CALL 15 SEEN TRUE
```

Call failure reasons

Code	Description	Possible reason
1	CALL 181 FAILUREREASON 1	Miscellaneous error
2	CALL 181 FAILUREREASON 2	User or phone number does not exist. Check that a prefix is entered for the phone number, either in the form 003725555555 or +3725555555; the form 3725555555 is incorrect.
3	CALL 181 FAILUREREASON 3	User is offline
4	CALL 181 FAILUREREASON 4	No proxy found
5	CALL 181 FAILUREREASON 5	Session terminated.
6	CALL 181 FAILUREREASON 6	No common codec found.

Code	Description	Possible reason
7	CALL 181 FAILUREREASON 7	Sound I/O error.
8	CALL 181 FAILUREREASON 8	Problem with remote sound device.
9	CALL 181 FAILUREREASON 9	Call blocked by recipient.
10	CALL 181 FAILUREREASON 10	Recipient not a friend.
11	CALL 181 FAILUREREASON 11	Current user not authorized by recipient.
12	CALL 181 FAILUREREASON 12	Sound recording error.

Leaving and manipulating voicemails

The commands to leave and manipulate voicemails are:

- Use the [VOICEMAIL command](#) to leave a voicemail for a target user.
- Use the [OPEN_VOICEMAIL command](#) to open and listen to voicemails.
- Use the [ALTER_VOICEMAIL command](#) to manipulate voicemails.

VOICEMAIL

To leave a voicemail:

Syntax

```
VOICEMAIL <target>
```

Response

```
VOICEMAIL <id> STATUS <value>
```

Parameters

- <target> - Skype username to receive the voicemail
- <value> - Refer to [voicemail object](#) for a list of available status values

Errors

Version

Protocol 5

Notes

- Leaving a voicemail for a target user actually uses two types of voicemail object:
 - a greeting type of voicemail object which is downloaded from the server
 - an outgoing type of voicemail object which the user composes

OPEN VOICEMAIL

To open and start playing a voicemail:

Syntax

```
OPEN VOICEMAIL <id>
```

Notes

- Voicemail is downloaded from server automatically.
- The main Skype window comes into focus and switches to the Call List tab; use the [ALTER VOICEMAIL command](#) to play without a UI response.

See also the [SEARCH VOICEMAILS](#) command.

ALTER VOICEMAIL

The ALTER VOICEMAIL command allows finer control over the VOICEMAIL object.

Syntax

```
ALTER VOICEMAIL <id> action
```

Response

```
ALTER VOICEMAIL action
```

Parameters

action, possible values:

- STARTPLAYBACK - starts playing downloaded voicemail
- STOPPLAYBACK - stops voicemail playback
- UPLOAD - uploads recorded voicemail from a local computer to a server
- DOWNLOAD - downloads voicemail object from a server to a local computer
- STARTRECORDING - stops playing greeting and starts recording, the equivalent to a user pressing the green button;
- STOPRECORDING - ends recording, the equivalent to a user pressing the red button
- DELETE - delete voicemail object

Notes

- STARTPLAYBACK plays voicemail but the window does not change to the Call List tab as it does with the OPEN VOICEMAIL command.
- STOPRECORDING causes automatic message upload to the server.
- Voicemails are deleted as a background process and the elapsed time depends on the server response; during this period, the SEARCH VOICEMAILS command still returns an ID for the voicemail, but the status is changed to DELETING.

Creating chats and sending messages

The commands for creating chats and sending messages are:

- Use the [CHAT CREATE command](#) to create a chat.
- Use the [CHATMESSAGE command](#) to send a chat message.
- Use the [ALTER_CHAT_SETTOPIC command](#) to set the topic for a chat.
- Use the [ALTER_CHAT_ADDMEMBERS command](#) to add members to a chat.
- Use the [ALTER_CHAT_LEAVE command](#) to leave a chat.
- Use the [GET_CHAT_CHATMESSAGES command](#) to retrieve the message identifiers for a chat.
- Use the [GET_CHAT_RECENTCHATMESSAGES command](#) to retrieve recent chat messages.
- Use the [SET_CHATMESSAGE_SEEN command](#) to mark a chat message as seen.
- The [SET_MESSAGE_SEEN command](#) is obsolete.
- The [MESSAGE command](#) is obsolete.

CHAT CREATE

Syntax

```
CHAT CREATE <target>[, <target>]*
```

Response

```
CHAT <chat_id> STATUS <value>
```

Version

Protocol 5

Parameters

- <target> - username(s) with whom to create a chat
- <chat_id> - chat identifier; string (usually looks like "#me/\$target;012345679012345")
- <value> - depends on the type of chat created: DIALOG for a 1:1 chat; MULTI_SUBSCRIBED for a chat with multiple participants

Notes

- The CHAT CREATE command does not open a chat window; use the [OPEN_CHAT](#) command to do so.

CHATMESSAGE

Syntax

```
CHATMESSAGE <chat_id> <message>
```

Response

```
CHATMESSAGE <id> STATUS SENDING
```

Parameters

- <chat_id> - chat identifier
- <message> - message text body to send
- <id> - chatmessage identifier

Version

Protocol 5

Errors

- ERROR 510 Invalid/unknown chat name given
Chat with <chat_id> does not exist
- ERROR 511 Sending a message to chat fails
Could not send message to chat (eg. not a member)

ALTER CHAT SETTOPIC

Syntax

```
ALTER CHAT <chat_id> SETTOPIC <topic>
```

Response

```
ALTER CHAT SETTOPIC
```

Version

Protocol 5

Errors

- ERROR 501 CHAT: No chat found for given chat
Chat with <id> does not exist

ALTER CHAT ADDMEMBERS

Syntax

```
ALTER CHAT <chat_id> ADDMEMBERS <target>[, <target>]*
```

Response

```
ALTER CHAT ADDMEMBERS
```

Version

Protocol 5

Errors

- ERROR 501 CHAT: No chat found for given chat
Chat with <chat_id> does not exist
- ERROR 504 CHAT: Action failed
Could not add members into chat (eg <target> is already a member; you have left chat)

ALTER CHAT LEAVE

Syntax

```
ALTER CHAT <chat_id> LEAVE
```

Response

```
ALTER CHAT LEAVE
```

Errors

- ERROR 501 CHAT: No chat found for given chat
Chat with <chat_id> does not exist
- ERROR 504 CHAT: Action failed
Could not leave chat (for example if the user has already left this chat)

GET CHAT CHATMESSAGES

Syntax

```
GET CHAT <chat_id> CHATMESSAGES
```

Response

```
CHAT <chat_id> CHATMESSAGES <id>[, <id>]*
```

Version

Protocol 5

Errors

- ERROR 501 CHAT: No chat found for given chat
Chat with <chat_id> does not exist

GET CHAT RECENTCHATMESSAGES

Syntax

```
GET CHAT <chat_id> RECENTCHATMESAGES
```

Response

```
CHAT <chat_id> RECENTCHATMESSAGES <id>[, <id>]*
```

Version

Protocol 5

Errors

- ERROR 501 CHAT: No chat found for given chat
Chat with <chat_id> does not exist

SET CHATMESSAGE SEEN

Syntax

```
SET CHATMESSAGE <id> SEEN
```

Response

```
CHATMESSAGE <id> STATUS <value>
```

Parameters

- <id> - chat message ID.
- <value> - new value for chat message status; refer to [CHATMESSAGE object](#) for status values

Version

Protocol 3

Example

```
-> SET CHATMESSAGE 61 SEEN
<- CHATMESSAGE 61 STATUS READ
```

Errors

- ERROR 18 SET: invalid WHAT

CHATMESSAGE command is missing or misspelled

- ERROR 31 Unknown message id
Unknown chat message ID
- ERROR 30 Invalid message id
Chat message ID is misspelled or contains non-permitted symbols (numeric are permitted)
- ERROR 32 Invalid WHAT
Invalid status given to chat message, for example the message is already marked as seen

SET MESSAGE SEEN- obsolete

Mark message as seen by the user and remove it from the missed messages list. This command is obsolete and has been replaced by the SET CHATMESSAGE SEEN command.

Syntax

```
SET MESSAGE <id> SEEN
```

Response

```
MESSAGE <id> STATUS value
```

Properties

- <id> - message ID;
- value - (new) status value

Version

Protocol 1, deprecated in protocol 3

Example

```
-> SET MESSAGE 1578 SEEN  
<- MESSAGE 1578 STATUS READ
```

Errors

- ERROR 18 SET: invalid WHAT
Object name missing or misspelled.
- ERROR 30 Invalid message id
ID includes other than numeric characters.
- ERROR 31 Unknown message id
Message with specified ID does not exist in current user's message history.
- ERROR 32 Invalid WHAT
Property name missing or misspelled.

MESSAGE - obsolete

The MESSAGE command is obsolete and has been replaced by the [CHATMESSAGE](#) command.

Syntax

```
MESSAGE <target> <text>
```

Response

```
CHATMESSAGE <id> STATUS SENDING(protocol 3 and up)
MESSAGE <id> STATUS SENDING(protocol 1 and 2)
```

Parameters

- `<target>` - target username to whom to send the message
- `<text>` - message body, for example Please call me

Version

Protocol 1

Errors

- ERROR 26 Invalid user handle
Target username missing or includes symbols which are not premitted
- ERROR 43 Cannot send empty message
Message has no body (e.g. "MESSAGE echo123")

Notes

When message sending fails, a LEFT-type message is received. The message's LEAVEREASON shows why it failed. See the [CHATMESSAGE object](#) for a description.

Example

```
-> MESSAGE echo123 Please call me
<- MESSAGE 982 STATUS SENDING
<- MESSAGE 982 STATUS SENT
```

Search commands

The search command requests specific information about objects. If no target is specified, all results for specified objects are returned.

Syntax

```
SEARCH USERS | FRIENDS | CALLS [<target>] | ACTIVECALLS |
MISSEDCALLS | VOICEMAILS | CHATS | MISSEDCHATS | ACTIVECHATS |
RECENTCHATS | BOOKMARKEDCHATS | CHATMESSAGES [<target>] |
MISSEDCHATMESSAGES | MESSAGES [<target>] | MISSEDMESSAGES |
USERSWAITINGMYAUTHORIZATION
```

Notes

- In Skype for Windows 1.1 only one search at a time is allowed; since version 1.2 multiple searches can be executed at the same time;
- The number of search results is not limited.
- SkypeOut contacts: since Skype for Windows 1.2 release it is possible to get the list of SkypeOut contacts which are part of the main contact list and they are returned with the contact list numbers, if the `SEARCH FRIENDS` command is executed. To get more information about the number in a current user's SkypeOut contacts use the `GET USER <number> <fullname>` command.

The search commands are:



- Use the [SEARCH FRIENDS command](#) to search for users in the contact list.
- Use the [SEARCH USERS command](#) to search for users.
- Use the [SEARCH CALLS command](#) to search for calls.
- Use the [SEARCH ACTIVECALLS command](#) to search for active calls.
- Use the [SEARCH MISSEDCALLS command](#) to search for missed calls.
- Use the [SEARCH VOICEMAILS command](#) to search for voicemails.
- The [SEARCH MESSAGES command](#) is obsolete.
- The [SEARCH MISSEDMESSAGES command](#) is obsolete. Use the to search for missed messages.
- Use the [SEARCH CHATS command](#) to search for chats.
- Use the [SEARCH ACTIVECHATS command](#) to search for active chats.
- Use the [SEARCH MISSEDCHATS command](#) to search for unread chat messages.
- Use the [SEARCH RECENTCHATS command](#) to search for recent chats.
- Use the [SEARCH BOOKMARKEDCHATS command](#) to search for bookmarked chats.
- Use the [SEARCH CHATMESSAGES command](#) to search for chatmessages.
- Use the [SEARCH MISSEDCHATMESSAGES command](#) to search for missed chat messages.
- Use the [SEARCH USERSWAITINGMYAUTHORIZATION command](#) to search for users awaiting authorization.

SEARCH FRIENDS

Syntax

```
SEARCH FRIENDS
```

Response

```
USERS [user[, user]*]  
returns a list of found usernames; an empty list if no match is found
```

Errors

- ERROR 67 target not allowed with SEARCH FRIENDS
a target (such as mike) was specified with the SEARCH FRIENDS command

Version

Protocol 1

Example

```
-> SEARCH FRIENDS  
<- USERS tim, joe, mike
```

SEARCH USERS

Syntax

```
SEARCH USERS <target>
```

Parameters

<target> - part of username or e-mail to match. If the search string contains "@", the search is performed by e-mail address and has to be an exact match. If the search string is a valid Skype username, the search is performed on the full name and username fields. In all other cases the search is made on the full name field only.

Response

```
SEARCH [<username>[, <username>]*]
```

returns a list of found usernames; list is empty if no match was found

Errors

- ERROR 4 Empty target not allowed
Target username is not specified

Notes

When running the `SEARCH USERS` command, `USER` notifications are reported back to the API client as users are found on the network. The API client should ignore these events and request each user's property after the search.

Version

Protocol 1

Example

```
-> #123 SEARCH USERS echo123
<- #123 USERS echo123, echo1232885
```

SEARCH CALLS

Syntax

```
SEARCH CALLS <target>
```

Parameters

<target> - username. Specifying a target is optional. If a target is specified, Skype searches the call history between the current user and the target user.

Response

```
CALLS [id[, id]*]
```

Returns a list of call IDs. If a target is specified, Skype returns IDs of all calls that have been made between the current and target user.

Errors

- ERROR 5 Search CALLS: invalid target
Characters that are not permitted were used in the target username. The username must have 6-22 characters and can contain only the following symbols: {a-Z0-9-_.}.

Version

Protocol 1

Example

```
-> SEARCH CALLS abc
<- CALLS 15, 16, 39
```

SEARCH ACTIVECALLS

Lists all calls visible on calltabs, including members of conference calls if the user is hosting a conference.

Syntax

```
SEARCH ACTIVECALLS
```

Response

```
CALLS [<id>[, <id>]*]
```

Returns a list of active call IDs.

Errors

- ERROR 3 SEARCH: unknown WHAT ACTIVECALLS was misspelled.

Version

Protocol 1

Example

```
-> SEARCH ACTIVECALLS
<- CALLS 25, 56
```

SEARCH MISSEDCALLS

Syntax

```
SEARCH MISSEDCALLS
```

Response

```
CALLS [<id>[, <id>]*]
```

Returns a list of missed call IDs, calls in MISSED status.

Errors

- ERROR 6 SEARCH MISSEDCALLS: target not allowed
No target is allowed with SEARCH MISSEDCALLS.

Version

Protocol 1

Example

```
-> SEARCH MISSEDCALLS
<- CALLS 25, 56
```

SEARCH VOICEMAILS

Syntax

```
SEARCH VOICEMAILS
```

Response

```
VOICEMAILS [<id>[, <id>]*]
```

Returns a list of voicemail IDs.||

Errors

- `ERROR 29 SEARCH VOICEMAILS: target not allowed`
No target is allowed with SEARCH VOICEMAILS

Version

Protocol 5

Example

```
->SEARCH VOICEMAILS  
<- VOICEMAILS 65, 70, 71
```

SEARCH MESSAGES

Syntax

```
SEARCH MESSAGES [<target>]
```

Parameters

- `<target>` - username. It is optional to specify a target. If a target is specified, Skype searches the message history between the current user and the target user.

Response

```
MESSAGES [<id>[, <id>]*]
```

Returns a list of message IDs. If a target is specified, Skype returns IDs of all messages that have been sent between the current user and the target user.

Errors

- `ERROR 5 SEARCH MESSAGES: invalid target`
A character was used in the target username that is not permitted. The username must have 6-22 characters and can contain only the following symbols: {a-Z0-9-_.}.

Version

Protocol 1, deprecated in protocol 3

Notes

This search is deprecated in protocol 3, use the [SEARCH CHATMESSAGES](#) command instead.

Example

```
-> SEARCH MESSAGES abc
<- MESSAGES 123, 124
```

SEARCH MISSEDMESSAGES

Syntax

```
SEARCH MISSEDMESSAGES
```

Response

```
MESSAGES [<id>[, <id>]*]
Returns a list of message IDs.
```

Errors

- ERROR 29 SEARCH MISSEDMESSAGES: target not allowed
No target is allowed with the SEARCH MISSEDMESSAGES command.

Version

Protocol 1, deprecated in protocol 3

Notes

This search is deprecated in protocol 3. Use the SEARCH MISSEDCCHATMESSAGES command instead.

Example

```
-> SEARCH MISSEDMESSAGES
<- MESSAGES 123, 124
```

SEARCH CHATS

Syntax

```
SEARCH CHATS
```

Response

```
CHATS [<chatname>[, <chatname>]*]
Returns a list of chat IDs.
```

Errors

- ERROR 107 target not allowed with CHATS
No target is allowed with the SEARCH CHATS command.

Version

Protocol 3

Example

```
-> SEARCH CHATS
<- CHATS #bitman/$jessy;eb06e65612353279,
#bitman/$jdenton;9244e98f82d7d391
```

SEARCH ACTIVECHATS

Syntax

```
SEARCH ACTIVECHATS
```

Response

```
CHATS [<chatname>[, <chatname>]*]
```

Returns a list of chat IDs that are open in the window.

Errors

- ERROR 29 No target allowed
No target is allowed with SEARCH ACTIVECHATS.

Version

Protocol 5

Example

```
-> SEARCH ACTIVECHATS
<- CHATS #bitman/$jessy;eb06e65612353279,
#bitman/$jdenton;9244e98f82d7d391
```

SEARCH MISSEDCHATS

Syntax

```
SEARCH MISSEDCHATS
```

Response

```
CHATS [<chatname>[, <chatname>]*]
```

Returns a list of chat IDs that include unread messages.

Errors

- ERROR 29 SEARCH MISSEDCHATS: target no allowed
No target is allowed with SEARCH MISSEDCHATS.

Version

Protocol 5

Example

```
-> SEARCH MISSEDCHATS
<- CHATS #bitman/$jessy;eb06e65612353279,
#bitman/$jdenton;9244e98f82d7d391
```

SEARCH RECENTCHATS

Syntax

```
SEARCH RECENTCHATS
```

Response

```
CHATS [<chatname>[, <chatname>]*]
```

Returns a list of recent chat IDs.

Errors

- ERROR 29 SEARCH RECENTCHATS: target no allowed
No target is allowed with SEARCH RECENTCHATS.

Version

Protocol 5

Example

```
-> SEARCH RECENTCHATS
<- CHATS #bitman/$jessy;eb06e65612353279,
#bitman/$jdenton;9244e98f82d7d391
```

SEARCH BOOKMARKEDCHATS

Syntax

SEARCH BOOKMARKEDCHATS

Response

CHATS [<chatname>[, <chatname>]*]

Returns a list of bookmarked chat IDs.

Errors

- ERROR 29 SEARCH BOOKMARKEDCHATS: target no allowed
No target is allowed with SEARCH BOOKMARKEDCHATS.

Version

Protocol 5

Example

```
-> SEARCH BOOKMARKEDCHATS
<- CHATS #bitman/$jessy;eb06e65612353279,
#bitman/$jdenton;9244e98f82d7d391
```

SEARCH CHATMESSAGES

Syntax

SEARCH CHATMESSAGES [<username>]

Parameters

<username> - target username, optional. If a username is specified, only chatmessages from/to that target user are returned.

Response

CHATMESSAGES [<id>[, <id>]*]

Returns a list of chat message IDs.

Errors

- ERROR 29 SEARCH CHATMESSAGES: invalid target
The target username contained a character that is not permitted.
(Username must have 6-22 characters and can contain only the following symbols: {a-Z0-9-_.}).

Version

Protocol 3

Example

```
-> SEARCH CHATMESSAGES abc
<- CHATMESSAGES 60, 59
```

SEARCH MISSEDCHATMESSAGES

Syntax

```
SEARCH MISSEDCHATMESSAGES
```

Response

```
CHATMESSAGES [<id>[, <id>]*]
```

Returns a list of missed chat message IDs.

Errors

- ERROR 29 SEARCH MISSEDCHATMESSAGES: target not allowed
No target is allowed with SEARCH MISSEDCHATMESSAGES.

Version

Protocol 3

Example

```
-> SEARCH MISSEDCHATMESSAGES  
<- CHATMESSAGES 61, 62
```

SEARCH USERSWAITINGMYAUTHORIZATION

Syntax

```
SEARCH USERSWAITINGMYAUTHORIZATION
```

Response

```
USERS [<id>[, <id>]*]
```

List of users who are waiting for authorization

Errors

- ERROR 29 SEARCH USERSWAITINGMYAUTHORIZATION: target not allowed

Version

Protocol 5

Example

```
-> SEARCH USERSWAITINGMYAUTHORIZATION  
<- USERS tim, john, echo123
```

Managing history

These commands are available to clear chat, voicemail, and call history.

CLEAR CHATHISTORY

Syntax

```
CLEAR CHATHISTORY
```

CLEAR VOICEMAILHISTORY

Syntax

CLEAR VOICEMAILHISTORY

CLEAR CALLHISTORY

Syntax

CLEAR CALLHISTORY

Controlling Skype user interface

This section lists the commands used to control the Skype user interface. The commands are:

- Use the [FOCUS command](#) to bring the Skype window into focus.
- Use the [MINIMIZE command](#) to minimize the Skype window.
- Use the [OPEN_ADDAFRIEND command](#) to open the Add a Contact window.
- The [OPEN_IM command](#) opens a chat window. However, this command has been superseded by the `OPEN_CHAT` command and new chat management commands in protocol 5.
- Use the [OPEN_CHAT command](#) to open a chat window.
- Use the [OPEN_FILETRANSFER command](#) to open a file transfer window.
- Use the [OPEN_VOICEMAIL command](#) to open a voicemail window.
- Use the [OPEN_PROFILE command](#) to open the profile for the current user.
- Use the [OPEN_USERINFO command](#) to open the profile window for a named Skype contact.
- Use the [OPEN_CONFERENCE command](#) to open the create conference window.
- Use the [OPEN_SEARCH command](#) to open a search window.
- Use the [OPEN_OPTIONS command](#) to open the options configuration window.
- Use the [OPEN_CALLHISTORY command](#) to open the call history tab in the main Skype window.
- Use the [OPEN_CONTACTS command](#) to open the contacts tab in the main Skype window.
- Use the [OPEN_DIALPAD command](#) to open the dialpad tab in the main Skype window.
- Use the [OPEN_SENDCONTACTS command](#) to open the send contacts window.
- Use the [OPEN_BLOCKEDUSERS command](#) to open the blocked users window.
- Use the [OPEN_IMPORTCONTACTS command](#) to open the import contacts window.
- Use the [OPEN_GETTINGSTARTED command](#) to open the getting started wizard.
- Use the [OPEN_AUTHORIZATION command](#) to open the authorization window.

FOCUS

The `FOCUS` brings the Skype window into focus on screen (on top).

Syntax

FOCUS

Response

If successful command is echoed back

Version

Protocol 1

MINIMIZE

This command minimizes the main Skype window into the system tray.

Syntax

MINIMIZE

Response

If successful command is echoed back

Version

Skype for Windows 1.3

Notes

This command does not minimize other Skype windows, such as chat or filetransfer.

OPEN ADDAFRIEND

This command opens the Add a Contact window.

Syntax

OPEN ADDAFRIEND [<username>]

Parameters

<username> - target username is optional. If a username is specified, the window is prefilled with it.

Response

If successful command is echoed back

Errors

ERROR 69 OPEN: invalid WHAT
Open target is missing or misspelled

Version

Skype for Windows 1.0

OPEN IM

This command opens the chat window with prefilled message.

Syntax

OPEN IM <username> [<message>]

Response

If successful command is echoed back

Parameters

- <username> - contact username to whom to send the message
- <message> - optional message body prefilled into the window

Errors

- ERROR 69 OPEN: invalid WHAT
Open target is missing or misspelled



- ERROR 70 Invalid user handle
Username is missing or contains not permitted symbols

Version

Skype for Windows 1.0

Notes

The protocol 5 chat management commands and Skype for Windows 1.3 `OPEN CHAT` command are preferable to the `OPEN IM` command.

Example

```
// IM dialog with echo123 will pop up, with text "This is a test!"  
already filled in as message text  
-> OPEN IM echo123 This is a test!  
<- OPEN IM echo123 This is a test!
```

OPEN CHAT

Syntax

```
OPEN CHAT <chat_id>
```

Parameters

<chat_id> - existing chat identifier (see `CHAT CREATE`, `SEARCH ...CHATS` commands)

Response

If successful command is echoed back

Errors

- ERROR 69 invalid open what
Open target is missing or misspelled
- ERROR 105 Invalid chat name
Chat id is missing or chat with this id doesn't exist.

Version

Skype for Windows 1.3

Example

```
-> OPEN CHAT #test/$echo123;52c2750d8686c10c  
<- OPEN CHAT #test/$echo123;52c2750d8686c10c
```

OPEN FILETRANSFER

Syntax

```
OPEN FILETRANSFER <username>[, <username>]*[ IN  
  <folder>]
```

Response

If successful command is echoed back

Parameters

- <username> - list of usernames to transfer file to;
- <folder> - optional, filesystem folder for file selection window. If not specified, the file transfer window opens in the default directory.

Errors

- ERROR 69 invalid open what
Open target is missing or misspelled
- ERROR 108 user not contact
Command is allowed for authorized contacts only
- ERROR 109 directory doesn't exist
Given folder does not exist or user has no access to it

Version

Skype for Windows 1.3

Example

```
-> OPEN FILETRANSFER echo123 IN C:\temp  
<- ERROR 108 user not contact  
-> OPEN FILETRANSFER myfriend IN C:\temp  
<- OPEN FILETRANSFER myfriend IN C:\temp
```

OPEN VOICEMAIL

Brings the callhistory tab into focus and starts playing a voicemail.

Syntax

```
OPEN VOICEMAIL <id>
```

Response

If successful command is echoed back

Parameters

<id> - voicemail identifier

Errors

- ERROR 69 invalid open what
Open target is missing or misspelled
- ERROR 512 invalid voicemail ID
Voicemail identifier is missing, is invalid or does not exist

OPEN PROFILE

This command opens the profile window for the current user.

Syntax

```
OPEN PROFILE
```

Response

If successful command is echoed back

Parameters**Errors**

- ERROR 69 invalid open what
Open target is missing or misspelled

Version

Skype for Windows 1.4

OPEN USERINFO

This command opens the profile window for a named Skype contact.

Syntax

```
OPEN USERINFO <skypename>
```

Response

If successful command is echoed back

Parameters

<skypename> - Skypename of contact

Errors

- ERROR invalid skypename
- ERROR 69 invalid open what
- Open target is missing or misspelled

Version

Skype for Windows 1.4

OPEN CONFERENCE

This command opens the create conference window.

Syntax

```
OPEN CONFERENCE
```

Response

If successful command is echoed back

Parameters

none

Errors

- ERROR 69 invalid open what
- Open target is missing or misspelled

Version

Skype for Windows 1.4

OPEN SEARCH

This command opens the search window.

Syntax

```
OPEN SEARCH
```

Response

If successful, command is echoed back

Parameters

none

Errors

- ERROR 69 invalid open what
- Open target is missing or misspelled

Version

Skype for Windows 1.4

OPEN OPTIONS

This command opens the options configuration window.

Syntax

```
OPEN OPTIONS <page>
```

Response

If successful, the command is echoed back

Parameters

<page>, possible values:

- general
- privacy
- notifications
- soundalerts
- sounddevices
- hotkeys
- connection
- voicemail
- callforward
- video (not yet implemented - will be added in Skype for Windows 2.0)
- advanced

Errors

- ERROR 69 invalid open what
- Open target is missing or misspelled

Version

Skype for Windows 1.4

Note: OPEN OPTIONS video parameter will be introduced in Skype for Windows 2.0.

OPEN CALLHISTORY

This command opens and sets the focus to the call history tab in the main Skype window.

Syntax

```
OPEN CALLHISTORY
```

Response

If successful, the command is echoed back

Parameters

none

Errors

- ERROR 69 invalid open what
- Open target is missing or misspelled

Version

Skype for Windows 1.4

OPEN CONTACTS

This command opens and sets the focus to the contacts tab in the main Skype window.

Syntax

```
OPEN CONTACTS
```

Response

If successful, the command is echoed back

Parameters

none

Errors

- ERROR 69 invalid open what
- Open target is missing or misspelled

Version

Skype for Windows 1.4

OPEN DIALPAD

This command opens and sets the focus to the dialpad tab in the main Skype window.

Syntax

```
OPEN DIALPAD
```

Response

If successful, the command is echoed back

Parameters

none

Errors

- ERROR 69 invalid open what
- Open target is missing or misspelled

Version

Skype for Windows 1.4

OPEN SENDCONTACTS

This command opens the send contacts window.

Syntax

```
OPEN SENDCONTACTS
```

Response

If successful, the command is echoed back

Parameters

none

Errors

- ERROR 69 invalid open what
- Open target is missing or misspelled

Version

Skype for Windows 1.4

OPEN BLOCKEDUSERS

This command opens the blocked users window.

Syntax

```
OPEN BLOCKEDUSERS
```

Response

If successful, the command is echoed back

Parameters

none

Errors

- ERROR 69 invalid open what
- Open target is missing or misspelled

Version

Skype for Windows 1.4

OPEN IMPORTCONTACTS

This command opens the import contacts window.

Syntax

```
OPEN IMPORTCONTACTS
```

Response

If successful, the command is echoed back

Parameters

none

Errors

- ERROR 69 invalid open what

- Open target is missing or misspelled

Version

Skype for Windows 1.4

OPEN GETTINGSTARTED

This command opens the getting started wizard.

Syntax

```
OPEN GETTINGSTARTED
```

Response

If successful, the command is echoed back

Parameters

none

Errors

- ERROR 69 invalid open what
- Open target is missing or misspelled

Version

Skype for Windows 1.4

OPEN AUTHORIZATION

This command opens the authorization request window for a given user.

Syntax

```
OPEN AUTHORIZATION <skypename>
```

Response

If successful, the command is echoed back

Parameters

`skypename` of the user whose authorization is requested

Errors

- ERROR 69 invalid open what
- Open target is missing or misspelled

Version

Skype for Windows 1.4

Application to application commands

The `AP2AP` feature in Skype allows two API clients to exchange information without the communication being visible on the client. Application to application communication has the following characteristics:

- Connections are only allowed to **connectable users**, online parties who are in the user's contact list or have active ongoing communication with the user.
- Connections are only attempted to connectable users at `CONNECT`.
- Connections are established only when there is a matching application

on the other side.

- The application name is limited to 32 bytes.
- Idle connections are dropped in a specific amount of time (typically 8 minutes).
- When connection is relayed, throttling is engaged.
- If the other party is logged in to multiple Skype instances, a stream for each instance is created.
- The stream write provides reliable transmission to deliver a large amount of data.
- The maximum amount of write to a stream can be 0xFFFF bytes long.
- Any character except 0x00 is allowed in a message.
- Datagrams are unreliable packets sent over a network (usually translates to UDP).
- The maximum size of datagrams is 1400 bytes.
- There is no guarantee that datagrams will be delivered.

The application to application commands are:

Use the [AP2AP_CREATE command](#) to register a new application object.

- Use the [AP2AP_CONNECT command](#) to create a stream from an application to a Skype user's instance of the application.
- Use the [AP2AP_WRITE command](#) to write text to the application stream.
- Use the [AP2AP_DATAGRAM command](#) to add a datagram to the application stream.
- Use the [AP2AP_READ command](#) to read data from the application stream.
- Use the [AP2AP_DISCONNECT command](#) to disconnect a user stream from an application.
- Use the [AP2AP_DELETE command](#) to delete an application and all connections to it.

Read an [application to application example](#) to get you started.

AP2AP_CREATE

This command registers a new application object with Skype.

Syntax

```
CREATE APPLICATION <appname>
```

Response

If successful, the command is echoed back

Parameters

<appname>: An arbitrary name to identify the application that will be exchanging data

Errors

- ERROR 536 CREATE: no object or type given
- ERROR 537 CREATE: Unknown object type given
- ERROR 540 CREATE APPLICATION: Missing or invalid name

Version

- Protocol 5

- Skype for Windows 1.4

AP2AP CONNECT

This command creates a stream from the application to another Skype user's instance of the same application.

Syntax

```
ALTER APPLICATION <appname> CONNECT <skypename>
```

Response

If successful, the command is echoed back

Parameters

- <appname>: An arbitrary name to identify the application that will be exchanging data
- <skypename>: The user to connect to this application

Errors

- ERROR 546 ALTER APPLICATION : Missing or invalid action
- ERROR 547 ALTER APPLICATION CONNECT: Invalid user handle

Version

- Protocol 5
- Skype for Windows 1.4

Note

If the user identified by <skypename> is logged in from multiple locations, a stream will be created to each location.

AP2AP WRITE

This command writes text into the application stream identified by the destination user's Skypename and stream ID.

Syntax

```
ALTER APPLICATION <appname> WRITE <skypename>:<id>  
<text>
```

Response

If successful, the command is echoed back

Note: There is a bug in Skype 1.4 where the number of SENDING notification following an application WRITE reports that the number of bytes sent is two characters longer than that which was written.

Parameters

- `<appname>`: An arbitrary name to identify the application that will be exchanging data
- `<skypename>`: The name of the skype contact to whom the message will be sent
- `<id>`: The numeric identifier for the skype instance to which the message will be sent
- `<text>`: The text to send

Errors

- ERROR 546 ALTER APPLICATION : Missing or invalid action
- ERROR 550 ALTER APPLICATION READ: Missing or Invalid stream identifier

Version

- Protocol 5
- Skype for Windows 1.4

Example

```
//Send "Hello world!" to user "testtest20" stream "1" (application "exe")
-> ALTER APPLICATION exe WRITE testtest20:1 Hello world!
<- ALTER APPLICATION exe WRITE testtest20:1
// The message has been queued for sending, message length is reported back
<- APPLICATION exe SENDING testtest20:1 14
// The message has been sent -- note missing stream ID from the end of response
<- APPLICATION exe SENDING

-> ALTER APPLICATION exe WRITE testtest20:1 1234567890
<- ALTER APPLICATION exe WRITE testtest20:1
<- APPLICATION exe SENDING testtest20:1 12
<- APPLICATION exe SENDING
```

AP2AP DATAGRAM

This command sends a datagram to the application stream.

Syntax

```
ALTER APPLICATION <appname> DATAGRAM <skypename>:<id>
<text>
```

Response

If successful, the command is echoed back

Parameters

`<appname>`: An arbitrary name to identify the application that will be exchanging data

Errors

- ERROR 546 ALTER APPLICATION : Missing or invalid

- action
- ERROR 551 ALTER APPLICATION DATAGRAM: Missing or invalid stream identifier

Version

- Protocol 5
- Skype for Windows 1.4

AP2AP READ

This command reads data from an application stream.

Syntax

```
ALTER APPLICATION <appname> READ <skypename>:<id>
```

Response

If successful, the command is echoed back

Parameters

<appname>: An arbitrary name to identify the application that will be exchanging data

Errors

- ERROR 546 ALTER APPLICATION : Missing or invalid action
- ERROR 550 ALTER APPLICATION READ: Missing or invalid stream identifier

Version

- Protocol 5
- Skype for Windows 1.4

AP2AP DISCONNECT

This command disconnects a user stream from an application.

Syntax

```
ALTER APPLICATION <appname> DISCONNECT <skypename>:<id>
```

Response

If successful, the command is echoed back

Parameters

- <appname>: An arbitrary name to identify the application that will be exchanging data
- <skypename>:<id>: The user and stream to disconnect

Errors

- ERROR 546 ALTER APPLICATION : Missing or invalid action
- ERROR 548 ALTER APPLICATION DISCONNECT: Invalid stream identifier

Version

- Protocol 5
- Skype for Windows 1.4

AP2AP DELETE

This command deletes an application and drops all connections to it.

Syntax

```
DELETE APPLICATION <appname>
```

Response

If successful, the command is echoed back

Parameters

<appname>: The name of the application to be deleted

Errors

- ERROR 538 DELETE: no object or type given
- ERROR 539 DELETE: Unknown object type given
- ERROR 542 DELETE APPLICATION : missing or invalid application name

Version

- Protocol 5
- Skype for Windows 1.4

Application to application example

Jim and Joe are two users who installed "toru" application.

```
// register application on both sides
[JIM] => CREATE APPLICATION toru
[JIM] <= CREATE APPLICATION toru

[JOE] => CREATE APPLICATION toru
[JOE] <= CREATE APPLICATION toru

// JIM initiates communication to JOE
[JIM] => ALTER APPLICATION toru CONNECT joe
[JIM] <= ALTER APPLICATION toru CONNECT joe

// connection establishing ...
[JIM] <= APPLICATION toru CONNECTING joe
// .. and is successful
[JIM] <= APPLICATION toru CONNECTING
// .. and creates one stream
[JIM] <= APPLICATION toru STREAMS joe:1

// and JOE is notified by new stream
[JOE] <= APPLICATION toru STREAMS jim:1

// JIM sends data over stream to JOE
[JIM] => ALTER APPLICATION toru WRITE joe:1 Hello world!
[JIM] <= ALTER APPLICATION toru WRITE joe:1
// stay tuned while data is transmitted...
[JIM] <= APPLICATION toru SENDING joe:1
// .. and you are notified on delivery success
[JIM] <= APPLICATION toru SENDING
```

```
// JOE receives notification about the incoming message
[JOE] <= APPLICATION toru RECEIVED jim:1
// .. and reads data from stream
[JOE] => ALTER APPLICATION toru READ jim:1
[JOE] <= ALTER APPLICATION toru READ jim:1 Hello world!
// ... and is notified that stream is empty
[JOE] <= APPLICATION toru RECEIVED

// JOE sends back acknowledgement of message
// A datagram is used because it is not so important to acknowledge
[JOE] => ALTER APPLICATION toru DATAGRAM jim:1 Hello back!
[JOE] <= ALTER APPLICATION toru DATAGRAM jim:1
// Now data is transmitted...
[JOE] <= APPLICATION toru SENDING jim:1=11
// .. and notified when it was sent (but delivery not assured)
[JOE] <= APPLICATION toru SENDING

// JIM receives datagram notification
[JIM] <= APPLICATION toru DATAGRAM joe:1 Hello back!

// JIM decides to end the communication
[JIM] => ALTER APPLICATION toru DISCONNECT joe:1
[JIM] <= ALTER APPLICATION toru DISCONNECT joe:1
// .. and when stream is closed it is notified
[JIM] <= APPLICATION toru STREAMS

// Also JOE receives notification that stream was closed
[JOE] <= APPLICATION toru STREAMS

// JIM unregisters applicaton
[JIM] => DELETE APPLICATION toru
[JIM] <= DELETE APPLICATION toru

// JOE unregisters applicaton
[JOE] => DELETE APPLICATION toru
[JOE] <= DELETE APPLICATION toru
```

Call forwarding

To set up call forwarding rules for an account:

Syntax

```
GET/SET PROFILE
{ CALL_NOANSWER_TIMEOUT <timeout>
| CALL_NOANSWER_ACTION {REJECT|FORWARD|VOICEMAIL} //
action if user does not answer
| CALL_FORWARD_RULES
[<start_time>,<end_time>,{<username>|<+PSTN>}[
<start_time>,<end_time>,{<username>|<+PSTN>}]*]
}
```

Parameters

- **timeout**- in seconds for call forward
- **start_time**- in seconds when connecting to this number/user starts
- **end_time** - in seconds when ringing to this number/user ends
- **username** - another Skype username to forward call
- **+PSTN** - PSTN number to forward a call



A call can be forwarded to multiple numbers and the numbers can overlap in time, with all ringing and the first to pick up the call takes it.

Version

Skype for Windows 1.4

Notes

- Skype Windows 1.4 UI defaults to no-answer timeout is 15 seconds
- Skype Windows 1.4 UI defaults to 0,60 start_time and end_time in rules
- Skype Windows 1.4 UI allows to set a maximum of 3 numbers for forwarded calls

Testing connections

To query the connection status:

Syntax

PING

Response

If successful PONG is echoed back

Version

Protocol 1

Objects

Skype has the following objects:

- [USER](#)
- [PROFILE](#)
- [CALL](#)
- [MESSAGE](#)
- [CHAT](#)
- [CHATMESSAGE](#)
- [VOICEMAIL](#)
- [APPLICATION](#)

USER

The user object has the following properties:

- **HANDLE** - username, for example: USER pamela HANDLE pamela.
- **FULLNAME** - user's full name, for example: USER pamela FULLNAME Jane Doe.
- **BIRTHDAY** - user's birth date in YYYYMMDD format, for example: USER bitman BIRTHDAY 19780329.
- **SEX** - **example:** USER pamela SEX UNKNOWN. **Values:**
 - UNKNOWN - user has not specified sex in personal profile.
- **MALE**
- **FEMALE**
- **LANGUAGE** - name of language, for example: USER mike LANGUAGE English. In protocol 4 with the ISO 639 prefix, example: USER mike LANGUAGE en English.
- **COUNTRY** - name of country, for example: USER mike COUNTRY

Estonia. In protocol 4 with the ISO 3166 prefix, example: USER mike COUNTRY ee Estonia.

- PROVINCE - **example:** USER mike PROVINCE Harjumaa.
- CITY - **example:** USER mike CITY Tallinn.
- PHONE_HOME - **example:** USER mike PHONE_HOME 3721111111.
- PHONE_OFFICE - **example:** USER mike PHONE_OFFICE 3721111111.
- PHONE_MOBILE - **example:** USER mike PHONE_MOBILE 3721111111.
- HOMEPAGE - **example:** USER mike HOMEPAGE http://www.joltid.com.
- ABOUT - **example:** USER mike ABOUT I am a nice person.
- HASCALEQUIPMENT - **returns always TRUE. Example:** USER pamela HASCALEQUIPMENT TRUE.
- BUDDYSTATUS - **example:** USER pamela BUDDYSTATUS 2. Possible BUDDYSTATUS values:
 - 0 - never been in contact list.
 - 1 - deleted from contact list. (read-write)
 - 2 - pending authorisation. (read-write)
 - 3 - added to contact list.
- ISAUTHORIZED - (read-write) is user authorized by current user? **Example:** USER pamela ISAUTHORIZED TRUE. Values:
 - TRUE
 - FALSE
- ISBLOCKED - (read-write) is user blocked by current user? **Example:** USER spammer ISBLOCKED TRUE. Values:
 - TRUE
 - FALSE
- DISPLAYNAME - **example:** USER pamela DISPLAYNAME pam.
- ONLINESTATUS - **user online status, for example:** USER mike ONLINESTATUS ONLINE. Possible values:
 - UNKNOWN - unknown user.
 - OFFLINE - user is offline (not connected). Will also be returned if current user is not authorized by other user to see his/her online status.
 - ONLINE - user is online.
 - AWAY - user is away (has been inactive for certain period).
 - NA - user is not available.
 - DND - user is in "Do not disturb" mode.
 - SKYPEOUT - user is in the SkypeOut contact list.
 - SKYPEME (Protocol 2)
 - LASTONLINETIMESTAMP - UNIX timestamp, available only for offline user. **Example** USER mike LASTONLINETIMESTAMP 1078959579.
 - CAN_LEAVE_VM - is it possible to send voicemail to user? **Example:** USER test CAN_LEAVE_VM TRUE. Possible values:
 - TRUE
 - FALSE
 - SPEEDDIAL - (read-write) speeddial code assigned to user.

- RECEIVEDAUTHREQUEST - text message for authorization request; available only when user asks for authorization.
- MOOD_TEXT - mood text for user (mood text is only visible to authorised users; visible in Skype for Windows 2.0).
- ALIASES <text> - list of assigned aliases (aliases are only visible as a result of a direct match for alias search).
- TIMEZONE <offset> - time offset from GMT in minutes; visible in Skype for Windows 2.0.
- IS_CF_ACTIVE - whether the user has Call Forwarding activated or not. Possible values:
 - TRUE
 - FALSE

Most user properties are read-only. The following properties are read-write and can be modified with the SET command:

- BUDDYSTATUS
 - 1 - delete from buddylist
- 2 - add user into contactlist and ask for authorization: SET USER echo123 BUDDYSTATUS 2 Please authorize me
- ISBLOCKED
 - TRUE - block user
- FALSE - unblock user
- ISAUTHORIZED
 - TRUE - authorize user
 - FALSE - dismiss authorization for user
- SPEEDDIAL - speeddial code assigned to user

PROFILE

Use the GET PROFILE command to retrieve profile information. The PROFILE object has the following properties:

- PSTN_BALANCE - **(read only)** SkypeOut balance value
- PSTN_BALANCE_CURRENCY - **(read only)** SkypeOut currency value
- FULLNAME - text
- BIRTHDAY - yyyyymmdd, 0 is returned if not set; no partial birthday allowed
- SEX - MALE | FEMALE | UNKNOWN
- LANGUAGES - [lang[lang]*] -- lang is a two letter ISO code (en, de, et)
- COUNTRY - iso2 name, a two letter ISO code; name - country name
- PROVINCE - text
- CITY - text
- PHONE_HOME - text
- PHONE_OFFICE - text
- PHONE_MOBILE - text
- HOMEPAGE - text
- ABOUT - text
- MOOD_TEXT - text
- TIMEZONE - offset is given in minutes from GMT

- `CALL_NOANSWER_TIMEOUT` Time out on call -- See [Call forwarding](#)
- `CALL_NOANSWER_ACTION` REJECT|FORWARD|VOICEMAIL -- See [Call forwarding](#)
- `CALL_FORWARD_RULES` See [Call forwarding](#)

CALL

The `CALL` object has the following properties:

- `TIMESTAMP` - time when call was placed (UNIX timestamp), for example
`CALL 17 TIMESTAMP 1078958218`
- `PARTNER_HANDLE` - for example `CALL 17 PARTNER_HANDLE mike`
- `PARTNER_DISPNAME` - for example `CALL 17 PARTNER_DISPNAME Mike Mann`
- `CONF_ID` - if `CONF_ID>0` the call is a conference call, for example: `CALL 17 CONF_ID 0`
- `TYPE` - call type, for example: `CALL 17 TYPE OUTGOING_PSTN`.

Possible values:

- `INCOMING_PSTN` - incoming call from PSTN
- `OUTGOING_PSTN` - outgoing call to PSTN
- `INCOMING_P2P` - incoming call from P2P
- `OUTGOING_P2P` - outgoing call to P2P
- `STATUS` - call status, for example: `CALL 17 STATUS FAILED`. Possible values:
 - `UNPLACED` - call was never placed
 - `ROUTING` - call is currently being routed
 - `EARLYMEDIA` - with pstn it is possible that before a call is established, early media is played. For example it can be a calling tone or a waiting message such as all operators are busy.
 - `FAILED` - call failed - try to get a `FAILUREREASON` for more information.
 - `RINGING` - currently ringing
 - `INPROGRESS` - call is in progress
 - `ONHOLD` - call is placed on hold
 - `FINISHED` - call is finished
 - `MISSED` - call was missed
 - `REFUSED` - call was refused
 - `BUSY` - destination was busy
 - `CANCELLED(Protocol 2)`
 - `VM_BUFFERING_GREETING` - voicemail greeting is being downloaded
 - `VM_PLAYING_GREETING` - voicemail greeting is being played
 - `VM_RECORDING` - voicemail is being recorded
 - `VM_UPLOADING` - voicemail recording is finished and uploaded into server
 - `VM_SENT` - voicemail has successfully been sent
 - `VM_CANCELLED` - leaving voicemail has been cancelled
 - `VM_FAILED` - leaving voicemail failed; check `FAILUREREASON`
- `FAILUREREASON` - example: `CALL 17 FAILUREREASON 1(numeric)`.
- `SUBJECT` - not used.
- `PSTN_NUMBER` - example: `CALL 17 PSTN_NUMBER 372123123`.

- DURATION - **example:** CALL 17 DURATION 0.
- PSTN_STATUS - **error string from gateway, in the case of a PSTN call, for example:** CALL 26 PSTN_STATUS 6500 PSTN connection creation timeout.
- CONF_PARTICIPANTS_COUNT - number of non-hosts in the case of a conference call. Possible values:
 - 0 - call is not a conference. For the host, CONF_PARTICIPANTS_COUNT is always 0.
- 1 - call is former conference
- 2, 3, 4 - call is conference
- CONF_PARTICIPANT *n* - the handle of the *n*th participant in a conference call, the call type and status and the displayname of participants who are not the host. For example: CALL 59 CONF_PARTICIPANT 1 echo123 INCOMING_P2P INPROGRESS Echo Test Service.
- VM_DURATION
- VM_ALLOWED_DURATION - maximum duration in seconds allowed to leave voicemail

Notes

- Status values for voicemails (VM_*) and VM_DURATION/VM_ALLOWED_DURATION apply to calls which are forwarded into voicemail. This feature was introduced in protocol 5.

Most call properties are read-only. The following properties are read-write and can be modified with the SET command:

- STATUS - for call control. Possible values:
 - ONHOLD - hold call
- INPROGRESS - answer or resume call
- FINISHED - hang up call
- SEEN - sets call as seen, so that a missed call is seen and can be removed from the missed calls list.
- DTMF - sends VALUE as DTMF. Permitted symbols in VALUE are: {0..9,#,*}.
- JOIN_CONFERENCE - joins call with another call into conference. VALUE is another call's ID.

MESSAGE

Version

Protocol 1, **deprecated in protocol 3** and replaced by the [CHATMESSAGE](#) object.

Properties

- TIMESTAMP - time when the message was sent (UNIX timestamp), for **example:** MESSAGE 21 TIMESTAMP 1078958218
- PARTNER_HANDLE - for **example** MESSAGE 21 PARTNER_HANDLE mike
- PARTNER_DISPNAME - for **example** MESSAGE 21 PARTNER_DISPNAME



Mike Mann

- `CONF_ID` - not used.
- `TYPE` - message type. for example `MESSAGE 21 TYPE TEXT`. Possible `TYPE` values:
 - `AUTHREQUEST` - authorization request
- `TEXT` - IM or topic set
- `CONTACTS` - contacts data
- `UNKNOWN` - other
- `STATUS` - message status, for example `MESSAGE 21 STATUS QUEUED`. Possible `STATUS` values:
 - `SENDING` - message is being sent
- `SENT` - message was sent
- `FAILED` - message sending failed. Try to get a `FAILUREREASON` for more information.
- `RECEIVED` - message has been received
- `READ` - message has been read
- `IGNORED` - message was ignored
- `QUEUED` - message is queued
- `FAILUREREASON` - for example `MESSAGE 21 FAILUREREASON 1(numeric)`.
- `BODY` - message body, for example `MESSAGE 21 BODY Hi, what's up?`

Most message properties are read-only. The following property is read-write and can be modified with the `SET` command:

- `SEEN` - the message is seen and will be removed from missed messages list. The UI sets this automatically if auto-popup is enabled for the user.

CHAT

Version

Protocol 3

Properties

- `NAME` - chat ID, for example `CHAT #test_1/$6a072ce5537c4044`
`NAME #test_1/$6a072ce5537c4044`
- `TIMESTAMP` - time when chat was created, for example `CHAT #test_1/$6a072ce5537c4044 TIMESTAMP 1078958218`
- `ADDER` - user who added the current user to chat, for example `CHAT 1078958218 ADDER k6rberebane`
- `STATUS` - chat status, for example `CHAT #test_1/$6a072ce5537c4044 STATUS MULTI_SUBSCRIBED`. Possible `STATUS` values:
 - `LEGACY_DIALOG` - old style IM
- `DIALOG` - 1:1 chat.
- `MULTI_SUBSCRIBED` - participant in chat
- `UNSUBSCRIBED` - left chat
- `POSTERS` - members who have posted messages, for example `CHAT #test_1/$6a072ce5537c4044 POSTERS k6rberebane test 3`

- **MEMBERS** - all users who have been there, for example `CHAT #test_1/$6a072ce5537c4044 MEMBERS k6rberebane test test_2 test_3`
- **TOPIC** - chat topic. Example: `CHAT #test_1/$6a072ce5537c4044 TOPIC API testimine`
- **CHATMESSAGES** - all messages IDs in this chat, for example `CHAT #test_1/$6a072ce5537c4044 CHATMESSAGES 34, 35, 36, 38, 39`
- **ACTIVEMEMBERS** - members who have stayed in chat, for example `CHAT #test_1/$6a072ce5537c4044 ACTIVEMEMBERS k6rberebane test_2 test_3`
- **FRIENDLYNAME** - name shown in chat window title, for example `CHAT #test_1/$6a072ce5537c4044 FRIENDLYNAME Test Test XX | tere ise ka`
- **CHATMESSAGES** - list of chatmessage identifiers
- **RECENTCHATMESSAGES** list of missed/recent chatmessage identifiers

CHATMESSAGE

Version

Protocol 3. Supersedes the MESSAGE object.

Properties

- **TIMESTAMP** - time when message was sent (UNIX timestamp), for example `MESSAGE 21 TIMESTAMP 1078958218`
- **PARTNER_HANDLE** - for example `CHATMESSAGE 21 PARTNER_HANDLE mike`
- **PARTNER_DISPNAME** - for example `CHATMESSAGE 21 PARTNER_DISPNAME Mike Mann`
- **TYPE** - message type, for example `MESSAGE 21 TYPE TEXT`. Possible TYPE values:
 - **SETTOPIC** - chat's topic change
- **SAID** - IM
- **ADDEDMEMBERS** - invited someone to chat
- **SAWMEMBERS** - chat participant has seen other members
- **CREATEDCHATWITH** - chat to multiple people is created
- **LEFT** - someone left chat; can also be a notification if somebody cannot be added to chat
- **UNKNOWN** - other
- **STATUS** - message status, for example `MESSAGE 21 STATUS QUEUED`. Possible STATUS values:
 - **SENDING** - message is being sent
- **SENT** - message was sent
- **RECEIVED** - message has been received
- **READ** - message has been read
- **LEAVEREASON** - used with LEFT type message, for example `CHATMESSAGE 21 LEAVEREASON UNSUBSCRIBE`. Possible LEAVEREASON values:
 - **USER_NOT_FOUND** - user was not found

- `USER_INCAPABLE` - user has an older Skype version and cannot join multichat
- `ADDER_MUST_BE_FRIEND` - recipient accepts messages from contacts only and sender is not in his/her contact list
- `ADDED_MUST_BE_AUTHORIZED` - recipient accepts messages from authorized users only and sender is not authorized
- `UNSUBSCRIBE` - participant left chat
- `BODY` - message body, for example `CHATMESSAGE 21 BODY Hi, what's up?`
- `CHATNAME` - chat that includes the message, for example `#test_3/$b17eb511457e9d20`
- `USERS` - people added to chat

Most chatmessage properties are read-only. The following property is read-write and can be modified with the `SET` command:

- `SEEN` - mark missed chatmessage as seen and removes chat from missed events.

VOICEMAIL

Version

Protocol 5

Properties

- `TYPE` - type of voicemail object
 - `INCOMING` - voicemail received from partner
- `OUTGOING` - voicemail sent to partner
- `DEFAULT_GREETING` - Skype default greeting from partner
- `CUSTOM_GREETING` - partner's recorded custom greeting
- `UNKNOWN`
- `PARTNER_HANDLE` - username for voicemail sender (for incoming) or recipient (for outgoing)
- `PARTNER_DISPNAME` - user displayname for partner
- `STATUS` - current status of voicemail object
 - `NOTDOWNLOADED` - voicemail is stored on server (has not been downloaded yet)
 - `DOWNLOADING` - downloading from server to local machine
 - `UNPLAYED` - voicemail has been downloaded but not played back yet
 - `BUFFERING` - buffering for playback
 - `PLAYING` - currently played back
 - `PLAYED` - voicemail has been played back
 - `BLANK` - intermediate status when new object is created but recording has not begun
 - `RECORDING` - voicemail currently being recorded
 - `RECORDED` - voicemail recorded but not yet uploaded to the server
 - `UPLOADING` - voicemail object is currently being uploaded to server
 - `UPLOADED` - upload to server finished but not yet deleted; object is also locally stored
 - `DELETING` - pending delete
 - `FAILED` - downloading voicemail/greeting failed



- UNKNOWN
- FAILUREREASON-
 - MISC_ERROR
- CONNECT_ERROR
- NO_VOICEMAIL_PRIVILEGE
- NO_SUCH_VOICEMAIL
- FILE_READ_ERROR
- FILE_WRITE_ERROR
- RECORDING_ERROR
- PLAYBACK_ERROR
- UNKNOWN
- SUBJECT - **not used**
- TIMESTAMP
- DURATION - **actual voicemail duration in seconds**
- ALLOWED_DURATION - **maximum voicemail duration in seconds allowed to leave to partner**

APPLICATION

Properties

- CONNECTABLE - query connectable users

```
-> GET APPLICATION appname CONNECTABLE
<- APPLICATION appname CONNECTABLE [username[ username]*]
```

- CONNECTING - query on-going connection process after the connection is established. Username is removed from CONNECTING list.

```
-> GET APPLICATION appname CONNECTING
<- APPLICATION appname CONNECTING [username[ username]*]
```

- STREAMS - query open streams (connections)

```
-> GET APPLICATION appname STREAMS
<- APPLICATION appname STREAMS [username:id[ username:id]*]
```

- SENDING - query if currently sending any data. After the data is sent, the stream name is removed from the SENDING list

```
-> GET APPLICATION appname RECIEVED
<- APPLICATION appname SENDING [username:id=bytes [username:id bytes]*]
```

- **Note:** In Skype 1.4x, the number of bytes reported by the SENDING notification following an APPLICATION WRITE is 2 bytes longer than that which was written.

```
-> alter application exe write testtest20:1 w
<- ALTER APPLICATION exe WRITE testtest20:1

<- APPLICATION exe SENDING testtest20:1 3

-> alter application exe write testtest20:1 1234567890
<- ALTER APPLICATION exe WRITE testtest20:1
<- APPLICATION exe SENDING testtest20:1 12
```



- RECEIVED - query if there is data waiting in received buffer. After the data is read from the stream, the stream name is removed from the RECEIVED list.

```
-> GET APPLICATION appname RECEIVED
<- APPLICATION appname SENDING [username:id=bytes [username:id
bytes]*]
```

- incoming datagram notification

```
<- APPLICATION appname DATAGRAM user:id text
```

Version

- Protocol 5
- Skype for Windows 1.4

Managing object properties

Three commands are available for retrieving and modifying object properties and general parameters:

- GET - general request command to retrieve object properties and general parameters
- SET - to set object properties and modify general parameters
- ALTER - to alter or perform an action with an object

General syntax

- GET USER <username> property
| CALL <id> property
| MESSAGE <id> property
| CHAT <id> property
| CHATMESSAGE <id> property
| VOICEMAIL <id> property
- SET USER <username> property <value>
| CALL <id> property <value>
| MESSAGE <id> property <value>
| CHAT <id> property <value>
| CHATMESSAGE <id> property <value>
| VOICEMAIL <id> property <value>

See the corresponding object information for available properties and property values:

- [call object](#)
- [user object](#)
- [profile object](#)
- [chat object](#)
- [chatmessage object](#)
- [voicemail object](#)
- [application object](#)

The commands for managing object properties are:

- Use the [GET USER command](#) to retrieve information about a user.

- Use the [SET USER command](#) to modify information about a user.
- Use the [GET CALL command](#) to retrieve information about a call.
- Use the [GET CHAT command](#) to retrieve information about a chat.
- Use the [GET CHATMESSAGE command](#) to retrieve information about a chatmessage.
- The [GET MESSAGE command](#) is deprecated and has been replaced by the `GET CHATMESSAGE` command.
- Commands for the `APPLICATION` object are described in the [application object](#) information.

GET USER

This command returns property values for a specified user.

Syntax

```
GET USER <username> property
```

Response

```
USER <username> property <value>
```

Parameters

- `<username>` - Skype username to retrieve property
- `property` - property name.
Available properties are: `HANDLE`, `FULLNAME`, `BIRTHDAY`, `SEX`, `LANGUAGE`, `COUNTRY`, `PROVINCE`, `CITY`, `PHONE_HOME`, `PHONE_OFFICE`, `PHONE_MOBILE`, `HOMEPAGE`, `ABOUT`, `HASCALLEQUIPMENT`, `BUDDYSTATUS`, `ISAUTHORIZED`, `ISBLOCKED`, `DISPLAYNAME`, `ONLINESTATUS`, `LASTONLINETIMESTAMP`, `CAN_LEAVE_VM`, `SPEEDDIAL`, `RECEIVEDAUTHREQUEST` (protocol 5). Refer to the [user object](#) information for more detail.

Version

Protocol 1

Errors

- `ERROR 7 GET: invalid WHAT`
Object name missing or misspelled
- `ERROR 10 Invalid prop`
ID and/or property missing or misspelled.
- `ERROR 8 invalid handle`
`USERNAME` missing or includes a not permitted character . Note: The `GET USER <target> ONLINESTATUS` command returns the response `OFFLINE` unless the current user is authorized by the target user to see his/her online status.

Example

```
-> GET USER pamela FULLNAME  
<- USER pamela FULLNAME Jane Doe
```


SET USER

Syntax

```
SET USER <target> ISAUTHORIZED TRUE|FALSE - allow/disable
target to see current user's userstatus
SET USER <target> ISBLOCKED TRUE|FALSE - block/unblock target
user
SET USER <target> BUDDYSTATUS 1 - remove target from contactlist
SET USER <target> BUDDYSTATUS 2 <message> - add target into
contactlist and ask authorization with message
```

GET CALL

This command returns property values for a specified call.

Syntax

```
GET CALL <id> property
```

Response

```
CALL <id> property <value>
```

Parameters

- <id> - call ID (numeric);
- property - property name.
Available properties are: `TIMESTAMP` (UNIX timestamp),
`PARTNER_HANDLE`, `PARTNER_DISPNAME`, `CONF_ID`, `TYPE`, `STATUS`,
`FAILUREREASON` (numeric), `SUBJECT` (not used),
`PSTN_NUMBER`, `DURATION`, `PSTN_STATUS`, `CONF_PARTICIPANT n`
(for non-host in conference only),
`CONF_PARTICIPANTS_COUNT`. Refer to the [call object](#) information for
more detail.

Version

Protocol 1

Errors

- ERROR 7 GET: invalid WHAT
Object name missing or misspelled.
- ERROR 11 Invalid call id
ID includes other than numeric characters.
- ERROR 12 Unknown call
Call with specified ID does not exist in current user's call history.
- ERROR 13 Invalid prop
Property name missing or misspelled.
- ERROR 71 Invalid conference participant NO
Conference participant's number is not a number or is too big

Example

```
-> GET CALL 1594 TYPE
<- CALL 1594 TYPE OUTGOING_P2P
```

GET CHAT

This command returns property values for a specified chat.

Syntax

```
GET CHAT <chat_id> property
```

Response

```
CHAT <chat_id> property <value>
```

Parameters

- <chat_id> - chat identifier;
- property - property name.
Available properties are: NAME, TIMESTAMP, ADDER, STATUS, POSTERS, MEMBERS, TOPIC, CHATMESSAGES, ACTIVEMEMBERS, FRIENDLYNAME.
See [CHAT object](#) description for detailed info.

Version

Protocol 3

Errors

- ERROR 7 GET: invalid WHAT
Object name missing or misspelled.
- ERROR 105 invalid chat name
Error in the CHATNAME parameter.
- ERROR 106 Invalid PROP
Property name missing or misspelled.

Example

```
-> GET CHAT #bitman/$jessy;eb06e65635359671 NAME
<- CHAT #bitman/$jessy;eb06e65635359671 NAME
#bitman/$jessy;eb06e65635359671
```

GET CHATMESSAGE

This command returns property values for a specified chat message.

Syntax

```
GET CHATMESSAGE <id> property
```

Response

```
CHATMESSAGE <id> property <value>
```

Parameters

- <id> - chat message ID;
- property - property name.

Available properties are: CHATNAME, TIMESTAMP, FROM_HANDLE, FROM_DISPNAME, TYPE, USERS, LEAVEREASON, BODY, STATUS. Refer to the [chatmessage object](#) information for more detail.

Version

Protocol 3

Example

```
-> GET CHATMESSAGE 60 CHATNAME  
<- CHATMESSAGE 60 CHATNAME #bitman/$jessy;eb06e65631239671
```

Errors

- ERROR 7 GET: invalid WHAT
Object name missing or misspelled.
- ERROR 14 Invalid message id
Chat message ID contains not permitted symbols (only numeric are permitted)
- ERROR 15 Unknown message
Unknown chat message ID
- ERROR 16 Invalid PROP
Property name missing or misspelled

GET MESSAGE

This command returns property values for a specified message. This command is deprecated since protocol 3, and was replaced by the [GET CHATMESSAGE](#) command.

Syntax

```
GET MESSAGE <id> property
```

Parameters

- <id> - chat message ID;
- property - property name.
Available properties are: TIMESTAMP (UNIX timestamp), PARTNER_HANDLE, PARTNER_DISPNAME, CONF_ID (not used), TYPE, STATUS, FAILUREREASON (numeric), BODY. Refer to the [message object](#) information for more detail.

Version

Protocol 1, **deprecated in protocol 3**

Errors

- ERROR 7 GET: invalid WHAT
Object name missing or misspelled.
- ERROR 14 Invalid message id
ID includes other than numeric characters.
- ERROR 15 Unknown message
Message with specified ID does not exist in current user's message

history.

- ERROR 16 Invalid prop
Property name missing or misspelled.

Example

```
-> GET MESSAGE 159 TYPE  
<- MESSAGE 159 TYPE TEXT
```

GET APPLICATION

For information about the GET APPLICATION command, refer to the [application object](#) information.

Managing general parameters

Use GET and SET commands to manage the following general variables:

- [GET SKYPEVERSION](#)
- [GET CURRENT USER](#)
- [GET USERSTATUS](#)
- [GET PRIVILEGE](#)
- [GET PROFILE](#)
- [GET CONNSTATUS \(connection\)](#)
- [AUDIO IN](#)
- [AUDIO OUT](#)
- [RINGER](#)
- [SET AVATAR](#)
- [RINGTONE](#)
- [RINGTONE STATUS](#)

GET SKYPEVERSION

Syntax

```
GET SKYPEVERSION
```

Response

```
SKYPEVERSION <version>
```

Version

Protocol 1

Example

```
-> GET SKYPEVERSION  
<- SKYPEVERSION 1.3.0.28
```

GET CURRENT USER

This command gets the username for the currently logged in user.

Syntax

```
GET CURRENTUSERHANDLE
```

Response

```
CURRENTUSERHANDLE <username>
```



Version

Protocol 1

GET USERSTATUS

This command queries or modifies user visibility for the current user.

Syntax

```
GET USERSTATUS
SET USERSTATUS <value>
```

Response

```
USERSTATUS <value>
```

Parameters

<value> - new userstatus. Possible values:

- UNKNOWN
- ONLINE - current user is online
- OFFLINE - current user is offline
- SKYPEME - current user is in "Skype Me" mode (protocol 2).
- AWAY - current user is away.
- NA - current user is not available.
- DND - current user is in "Do not disturb" mode.
- INVISIBLE - current user is invisible to others.
- LOGGEDOUT - current user is logged out. Clients are detached.

Version

Protocol 1

Errors

- ERROR 28 Unknown userstatus
Status value is incorrect or misspelled

Example

```
-> SET USERSTATUS OFFLINE
<- USERSTATUS OFFLINE
<- USERSTATUS OFFLINE
-> SET USERSTATUS xxx
<- ERROR 28 Unknown userstatus
```

GET PRIVILEGE

Syntax

```
GET PRIVILEGE user_privilege
```

Response

```
PRIVILEGE user_privilege <value>
```

Parameters

- user_privilege - possible values:

- SKYPEOUT
- SKYPEIN
- VOICEMAIL
- <value> - possible values:
 - TRUE - privilege/service is enabled
 - FALSE - privilege/service is disabled

Errors

- ERROR 40 Unknown privilege
Privilege name is missing or misspelled

Version

Protocol 1

Example

```
-> GET PRIVILEGE SKYPEOUT
<- PRIVILEGE SKYPEOUT TRUE
-> GET PRIVILEGE SKYPEIN
<- PRIVILEGE SKYPEIN FALSE
```

GET PROFILE

This command queries the current user's profile information.

Syntax

- GET PROFILE <profile_property>
- SET PROFILE <profile_property> <value>

Response

PROFILE profile_property <value>

Parameters

Possible values for each profile_property

- PSTN_BALANCE - **(Read only)** SkypeOut balance value
- PSTN_BALANCE_CURRENCY - **(Read only)** currency value
- FULLNAME - text
- BIRTHDAY - yyyymmdd, 0 is returned if not set; no partial birthday allowed
- SEX - MALE | FEMALE | UNKNOWN
- LANGUAGES - [lang[lang]*] -- lang is a two letter ISO code (en, de, et)
- COUNTRY - iso2 name -- iso2 is a two letter ISO code; name - country name
- PROVINCE - text
- CITY - text
- PHONE_HOME - text
- PHONE_OFFICE - text
- PHONE_MOBILE - text
- HOMEPAGE - text
- ABOUT - text



- MOOD_TEXT - text
- TIMEZONE - offset is given in minutes from GMT
- CALL_NOANSWER_TIMEOUT Time out on call - See [Call forwarding](#)
- CALL_NOANSWER_ACTION REJECT|FORWARD|VOICEMAIL - See [Call forwarding](#)
- CALL_FORWARD_RULES - See [Call forwarding](#)

Version

- Protocol 3
- Skype for Windows 1.4: FULLNAME, BIRTHDAY, SEX, LANGUAGES, COUNTRY, PROVINCE, CITY, PHONE_HOME, PHONE_OFFICE, PHONE_MOBILE, HOMEPAGE, ABOUT, MOOD_TEXT, TIMEZONE.
- MOOD_TEXT and TIMEZONE will be visible in Skype for Windows 2.0 only.

Notes

The current client allows you to set one language only.

Example

```
-> GET PROFILE PSTN_BALANCE
<- PROFILE PSTN_BALANCE 5000
-> GET PROFILE PSTN_BALANCE_CURRENCY
<- PROFILE PSTN_BALANCE_CURRENCY EUR
```

GET CONNSTATUS (connection)

This command returns the current network connection status.

Syntax

```
GET CONNSTATUS
```

Response

```
CONNSTATUS <value>
```

Parameters

<value> - possible values:

- OFFLINE
- CONNECTING
- PAUSING
- ONLINE

Version

Protocol 1

Example

```
-> GET CONNSTATUS
<- CONNSTATUS ONLINE
```

AUDIO IN

The `GET` command returns the current audio input device for Skype.
The `SET` command assigns a new audio input device for Skype.

Syntax

```
GET AUDIO_IN
SET AUDIO_IN <device_name>
```

Response

```
AUDIO_IN <device_name>
```

Version

Protocol 1

Note

Setting a device with an empty name selects the Windows default device.

Example

```
-> GET AUDIO_IN
<- AUDIO_IN SB Audigy 2 ZS Audio [DC00]
```

AUDIO OUT

The `GET` command returns the current audio output device for Skype.
The `SET` command assigns a new audio output device for Skype.

Syntax

```
GET AUDIO_OUT
SET AUDIO_OUT <device_name>
```

Response

```
AUDIO_OUT <device_name>
```

Version

Protocol 1

Note

Setting a device with an empty name selects the Windows default device.

Example

```
-> GET AUDIO_OUT
<- AUDIO_OUT SB Audigy 2 ZS Audio [DC00]<
```


RINGER

The `GET` command returns the current ringing device for Skype.
The `SET` command assigns a new ringing device for Skype.

Syntax

```
GET RINGER
SET RINGER <device_name>
```

Response

```
RINGER <device_name>
```

Version

Skype for Windows 1.3

Note

Setting a device with an empty name selects the Windows default device.

Example

```
-> GET RINGER
<- RINGER SB Audigy 2 ZS Audio [DC00]
```

MUTE

This command gets or sets the mute status.

Syntax

```
GET MUTE
SET MUTE ON|OFF
```

Response

```
MUTE ON|OFF
```

Version

Protocol 1

Notes

If there are currently no active calls (call status `INPROGRESS`), `MUTE` is always `OFF` and setting `MUTE ON` has no effect.

Example

```
-> GET MUTE
<- MUTE OFF
// set mute when no call is active - mute remains OFF
-> SET MUTE ON
<- MUTE OFF
```

SET AVATAR

This command changes the avatar picture for the user profile.

Syntax

```
SET AVATAR <id> <filePath + fileName>[:idx]
```

Response

```
AVATAR <id> <filePath + fileName>
```

Parameters

- `id` - avatar id. Currently `<id>` always set to '1' as only one default avatar is supported.
- `filePath` - avatar file directory.
- `fileName:idx` - avatar file may either be image or .skype file format. IDX refers to the content number in .skype file formats (0,..)

Version

- Skype for Windows 1.3
- .skype files are supported in Skype for Windows 1.4

Errors

- ERROR 114 Invalid avatar
Avatar id is missing or invalid
- ERROR 111 File not found
Avatar file specified does not exist
- ERROR 9901 internal error
Wrong type of file (for example an audio file or a document) is set to avatar

Example

```
-> SET AVATAR 1 C:\Documents and Settings\Administrator\My
Documents\My Pictures\kitten.jpg
<- AVATAR 1 C:\Documents and Settings\Administrator\My Documents\My
Pictures\kitten.jpg
```

RINGTONE

The `GET` command returns the current ringtone file for Skype.

The `SET` command assigns a new ringtone for Skype.

Syntax

- `GET RINGTONE <id>`
- `SET RINGTONE <id> <filePath + fileName>[:idx]`

Response

```
RINGTONE <id> <filePath + fileName>
```

Parameters

- `id` - ringtone id. Currently `<id>` should always be set to '1'
 - 1 - Default ringtone
- 1101 - Default ringtone
- 1102 - Ringback tone
- 1103 - Busy tone
- 1104 - Dialing tone
- 1105 - Connecting sound
- 1202 - Resume sound
- 1203 - Hangup sound

- 1204 - Incoming message sound
- 1205 - Online alert sound
- filePath - ringtone file directory.
- fileName:idx - ringtone file may either be .wav or .skype file format.
IDX refers to the content number in .skype file formats (0,..)

Version

- Skype for Windows 1.3
- .skype files are supported since Skype for Windows 1.4
- Querying ringtone status is supported since Skype for Windows 1.4

Errors

- ERROR 115 Invalid ringtone
Ringtone id is missing or invalid
- ERROR 111 File not found
Ringtone file specified does not exist

Notes

- If the Skype default ringtone is used, the GET command returns its name with no filepath.
- .skype may be used instead of .wav files and can contain multiple contents enumerated by integer IDs (idx).

Example

```
-> GET RINGTONE 1
<- RINGTONE 1 call_in
-> SET RINGTONE 1 C:/WINDOWS/Media/tada.wav
<- RINGTONE 1 C:/WINDOWS/Media/tada.wav
```

RINGTONE STATUS

This command queries if ringtones are enabled

Syntax

```
GET RINGTONE <id> STATUS
```

Response

```
RINGTONE <id> <ON|OFF>
```

Parameters

See above

Notifications

Notifications are sent by Skype if an object changes or if the value of a property is requested with a `GET` command. Also, if a property value is changed by a `SET` command, the change is confirmed with a notification. Notifications occur in the same manner, whether the related change is initiated by the Skype UI or by an API client. There are two main types of notification:

- [Object notifications](#) occur when an object is created (for example due to an incoming call or chat), if an object changes, or if a property is queried.
- [Status notifications](#) are broadcast by Skype after an initial connection is made or if a parameter changes. These notifications can be queried at any time with the `GET` command.

Skype object notifications

Skype object notifications are:

- [Call notifications](#)
- [User notifications](#)
- [Chat notifications](#)
- [Chatmessage notification](#)
- [Voicemail notifications](#)
- [Application notifications](#)

Call notifications

Call notifications are sent on incoming calls or when an active calls changes. Clients can monitor call events to detect incoming calls and act on them (for example, to answer automatically).

Syntax

```
CALL <id> property <value>
```

Parameters

Refer to the [call object](#) for available properties and property values.

User notifications

User notifications are the most frequent notifications and include last-seen timestamps and user property information.

Syntax

```
USER <id> property <value>
```

Parameters

Refer to the [user object](#) for available properties and property values.

Note

User notifications are reported also for users who are not in the contactlist. The client can ignore these events as they result from an unrelated command or event.

Chat notifications

Chat notification is sent when a chat is created, chat properties or members change, or a new message is posted into chat. A new message also triggers a chatmessage notification.

Syntax

```
CHAT <id> property <value>
```

Parameters

Refer to the [chat object](#) for available properties and property values.

Chatmessage notifications

Chatmessage notification is sent when a new message arrives. The client can monitor these messages to display received messages.

Syntax

```
CHATMESSAGE <id> property <value>  
MESSAGE <id> property <value>
```

Parameters

Refer to the [chatmessage object](#) for available properties and property values.

Notes

The MESSAGE command is deprecated in Protocol 3

Voicemail notifications

Voicemail notification is sent when a new voicemail is received or recorded.

Syntax

```
VOICEMAIL <id> property <value>
```

Parameters

Refer to the [voicemail object](#) for available properties and property values.

Application notifications

Application notifications are sent when a new connection requests to connect, or when data is sent or received.

Syntax

```
APPLICATION <appname> property <value>
```

Parameters

Refer to the [Application object](#) for available properties and property values.

Status notifications

Skype status notifications are:

- [Callhistory change notification](#)
- [Instant message history change](#)
- [Contactlist change notification](#)
- [User status notification](#)
- [Connection status](#)
- [Current user handle](#)
- [Contact list focus notification](#)

Callhistory change notification

This notification occurs when call history changes and needs to be reloaded. This change occurs when the call history or a selection of it has been deleted.

Syntax

```
CALLHISTORYCHANGED
```

Instant message history change

This notification occurs when instant message history changes and needs to be reloaded. It occurs only when all IM history is deleted.

Syntax

```
IMHISTORYCHANGED
```

Contactlist change notification

This notification occurs if a user is added to or deleted from contacts or has authorized the current user as a contact.

Syntax

```
USER <username> BUDDYSTATUS <status>
```

Parameters

Refer to the [user object](#) for available status values.

Example

```
// User has been added to contacts, pending authorisation.  
<- USER pamela BUDDYSTATUS 2  
// User has authorized current user  
<- USER pamela BUDDYSTATUS 3  
// User has been deleted from contacts.  
<- USER pamela BUDDYSTATUS 1
```

User status notification

Syntax

```
USERSTATUS status
```

Parameters

status - value for user status. Possible values:

- UNKNOWN - no status information for current user.
- ONLINE - current user is online.
- OFFLINE - current user is offline.

- SKYPEME - current user is in "Skype Me" mode (Protocol 2).
- AWAY - current user is away.
- NA - current user is not available.
- DND - current user is in "Do not disturb" mode.
- INVISIBLE - current user is invisible to others.
- LOGGEDOUT - current user is logged out. Clients are detached.

Connection status

Syntax

```
CONNSTATUS status
```

Parameters

`status` - value for connection status. Possible values:

- OFFLINE
- CONNECTING
- PAUSING
- ONLINE
- LOGGEDOUT - current user is logged out.

Current user handle

Syntax

```
CURRENTUSERHANDLE <username>
```

Example

```
CURRENTUSERHANDLE banana
```

Contact list focus notification

This notification occurs when contactlist focus changes:

Syntax

- CONTACTS FOCUSED `username` - when contact gains focus
- CONTACTS FOCUSED - when loses focus

Error codes

```
ERROR CODE [DESC]
```

Skype sends an error response when it encounters an issue such as incorrect commands or internal inconsistencies. The error code is a number that uniquely identifies the error condition and the `DESC` is an optional brief description of the issue.

Currently the following error codes are defined:

Code	Description	Possible reasons
1	General syntax error	Command missing (e.g. " " sent as command)
2	Unknown command	Command spelled incorrect (e.g. "GRT" send instead of "GET")
3	Search: unknown WHAT	Search target is missing or misspelled
4	Empty target not allowed	
5	Search CALLS: invalid target	Not permitted character (e.g. "!", "#", "\$", "€", " " (space) etc.) was used in the target username (e.g. "SEARCH CALLS !a")
6	SEARCH MISSEDCALLS: target not allowed	e.g. "SEARCH MISSEDCALLS echo123"
7	GET: invalid WHAT	Object/property name missing or misspelled
8	Invalid user handle	USERNAME missing or includes a not permitted character (e.g. "GET USER ! HANDLE")
9	Unknown user	
10	Invalid PROP	Property name and/or ID missing or misspelled
11	Invalid call id	Call ID missing or misspelled (must be a numeric value)
12	Unknown call	Nonexistant call ID used
13	Invalid PROP	Returned to command GET CALL id PARTNER_DISPLAYNAME. Property name missing or misspelled
14	Invalid message id	GET - Message ID missing or misspelled (must be a numeric value)
15	Unknown message	Nonexistant message ID used in GET command
16	Invalid PROP	Returned to command GET MESSAGE id PARTNER_DISPLAYNAME. Property name missing or misspelled
17	(Not in use)	
18	SET: invalid WHAT	Property name missing or misspelled
19	Invalid call id	Call ID missing or misspelled (must be a numeric value)
20	Unknown call	Nonexistant call ID used
21	Unknown/disallowed call prop	SET CALL value incorrect or misspelled (e.g. "SET CALL 15 STATUS ONHOL")
22	Cannot hold this call at the moment	Trying to hold a call that is not in progress.
23	Cannot resume this call at	Trying to resume/answer a call that is not in

Code	Description	Possible reasons
	the moment	progress.
24	Cannot hangup inactive call	Trying to hang up a call that is not in progress.
25	Unknown WHAT	Property name missing or misspelled (e.g. "SET CALL 15 STATU ONHOLD")
26	Invalid user handle	Target username missing or includes not permitted symbols (e.g. "MESSAGE ")
27	Invalid version number	Invalid protocol number (e.g. "PROTOCOL -12,9")
28	Unknown userstatus	Unknown or misspelled value for user status (e.g. "SET USERSTATUS RICH")
29	SEARCH what: target not allowed	Target is not permitted; e.g. "SEARCH MISSEDMESSAGES echo123"
30	Invalid message id	SET - Message ID missing or misspelled (must be a numeric value)
31	Unknown message id	Nonexistant message ID used in SET command
32	Invalid WHAT	Property missing or misspelled
33	invalid parameter	Unknown or misspelled value for mute (e.g. "SET MUTE O")
34	invalid user handle	Target username/number missing (e.g. "CALL ")
35	Not connected	
36	Not online	
37	Not connected	
38	Not online	
39	user blocked	Destination user is blocked by caller. Also given, if trying to call to a blocked user
40	Unknown privilege	Privilege is either misspelled or does not exist (e.g. "GET PRIVILEGE SKYPEOUT").
41	Call not active	Trying to send DTMF, when call is not active.
42	Invalid DTMF code	Invalid DTMF code is sent. Valid symbols for DTMF codes are {0..9,#,*}
43	cannot send empty message	Empty message is tried to sent, e.g. "MESSAGE echo123".
50	cannot set device	An error occurred when changing audio device
51	invalid parameter	Parameter to READY command is not YES or NO
52	invalid parameter	Parameter to HOOK command is not ON or OFF
53	invalid value	Parameter to SET AUTOAWAY is not ON or OFF
66	Not connected	Skype is not connected i.e. user status is

Code Description		Possible reasons
		"LOGGEDOUT"
67	Target not allowed with SEARCH FRIENDS	SEARCH FRIENDS had a parameter
68	Access denied	
69	Invalid open what	OPEN command had missing or misspelled TARGET e.g. "OPEN IN"
70	Invalid handle	OPEN IM parameter USERNAME is missing or contains not permitted symbols
71	Invalid conference participant NO	Conference participant's number is either too large or invalid.
72	Cannot create conference	
73	too many participants	Conference is initiated to more than 4 people.
74	Invalid key	Key name in BTN_PRESSED or BTN_RELEASED command is invalid
91	call error	Cannot call an emergency number
92	call error	The called number is not a valid PSTN number
93	call error	Invalid Skype Name
94	call error	Cannot call yourself
95	Internal error	Destination user is blocked by caller right after call initialization
96	Internal error	An outgoing call exists in ROUTING/RINGING/EARLYMEDIA state
97	Internal error	Internal error
98	Internal error	Internal error
99	Internal error	Internal error
100	Internal error	Internal error
101	Internal error	A call to the destination user is already ongoing
103	Cannot hold	Internal error
104	Cannot resume	Internal error
105	Invalid chat name	Chat name missing or misspelled
106	Invalid PROP	Property name missing or misspelled for CHAT or CHATMESSAGE
107	Target not allowed with CHATS	No parameters allowed to SEARCH CHATS
108	User not contact	TRANSFER can only be initiated to contacts
109	directory doesn't exist	Directory given as a parameter to TRANSFER command does not exist

Code	Description	Possible reasons
110	No voicemail capability	User given as a parameter to VOICEMAIL command doesn't have voicemail capability
111	File not found	File given as argument to SET AVATAR or SET RINGTONE command doesn't exist
112	Too many targets	Number of target users for OPEN FILETRANSFER command exceeds simultaneous filetransfer limit
113	Close: invalid WHAT	Invalid argument to CLOSE command
114	Invalid avatar	GET or SET AVATAR avatar index invalid
115	Invalid ringtone	GET or SET RINGTONE ringtone index invalid
500	CHAT: Invalid chat name given	
501	CHAT: No chat found for given chat	
502	CHAT: No action name given	
503	CHAT: Invalid or unknown action	
504	CHAT: action failed	
505	CHAT: LEAVE does not take arguments	
506	CHAT: ADDMEMBERS: invalid/missing user handle(s) as arguments	
507	CHAT: CREATE: invalid/missing user handle(s) as argument	
508	CHAT: CREATE: opening a dialog to the given user failed	
509	No chat name given	
510	Invalid/unknown chat name given	
511	Sending a message to chat fails	
512	Invalid voicemail id	
513	Invalid voicemail object	
514	No voicemail property given	
515	Assigning speeddial property failed	

Code	Description	Possible reasons
516	Invalid value given to ISAUTHORIZED/ISBLOCKED	
517	Changing ISAUTHORIZED/ISBLOCKED failed	
518	Invalid status given for BUDDYSTATUS	
519	Updating BUDDYSTATUS failed	
520	CLEAR needs a target	
521	Invalid/unknown CLEAR target	
522	CLEAR CHATHISTORY takes no arguments	
523	CLEAR VOICEMAILHISTORY takes no arguments	
524	CLEAR CALLHISTORY: missing target argument	
525	CLEAR CALLHISTORY: invalid handle argument	
526	ALTER: no object type given	
527	ALTER: unknown object type given	
528	VOICEMAIL: No proper voicemail ID given	
529	VOICEMAIL: Invalid voicemail ID given	
530	VOICEMAIL: No action given	
531	VOICEMAIL: Action failed	
532	VOICEMAIL: Unknown action	
534	SEARCH GREETING: invalid handle	
535	SEARCH GREETING: unable to get greeting	
536	CREATE: no object type given	

Code	Description	Possible reasons
537	CREATE : Unknown object type given.	
538	DELETE : no object type given.	
539	DELETE : unknown object type given.	
540	CREATE APPLICATION : missing or invalid name.	
541	APPLICATION : Operation Failed.	
542	DELETE APPLICATION : missing or invalid application name.	
543	GET APPLICATION : missing or invalid application name.	
544	GET APPLICATION : missing or invalid property name.	
545	ALTER APPLICATION : missing or invalid action.	
546	ALTER APPLICATION : Missing or invalid action	
547	ALTER APPLICATION CONNECT: Invalid user handle	
548	ALTER APPLICATION DISCONNECT: Invalid stream identifier	
549	ALTER APPLICATION WRITE : Missing or invalid stream identifier	
550	ALTER APPLICATION READ : Missing or invalid stream identifier	
551	ALTER APPLICATION DATAGRAM : Missing or invalid stream identifier	
552	SET PROFILE : invalid property profile given	
553	SET PROFILE CALL_SEND_TO_VM : no voicemail privledge,	

Code Description Possible reasons

can't forward to voicemail.

9901 Internal error

Skype URI handler

Although not part of the Skype public API, Skype 1.4 a number of commands which can be initiated using the skype URI handler.

General syntax

```

SKYPE_URI      = "skype:" [targets] ["?" query ] ["#" fragment ]
targets        = 1* (target / ";" )
target         = identity / PSTN
identity       = skypeName / alias
skypeName      = 1*(ALPHA / DIGIT / "." / "," )
skypeNames     = 1*( skypeName / ";" )
alias          = ... ; see ["TechGroup/DataFormats"]
                ; unicode chars are in UTF-8 and % encoded; see
RFC3987 uchar mapping
PSTN           = "+" (DIGIT / ALPHA ) *(DIGIT / ALPHA / "-" ) ;
supports +800-FLOWERS
query          = action [ *( "?" term "=" conditon ) ]
term           = 1*ALPHA
condition      = 1*unserved ; to be clarified
fragment       = 1*unserved ; to be clarified

```

Skype for Windows 1.4 version handles the following

```

skype:                ; focus / open skype UI
skype:[targets]       ; take default double-click action on
contact
skype:[targets]?call   ; call to target(s): can be
skypeName, alias or PSTN
skype:[skypeNames]?chat ; start chat/multichat with
skypeName(s)
skype:[skypeName]?voicemail ; leave voicemail to skypeName
skype:[skypeName]?add    ; add skypeName to contactlist; show
authorization dialog
skype:[skypeNames]?sendfile ; open sendfile dialog to skypeNames
skype:[skypeName]?userinfo ; show info (profile) for [username]
skype:?chat&id=[id][#time] ; open existing multichat with [id];
                        ; time: YYYY-MM-DDThh:mm:ssTZ / YYYY-
MM-DDZhh:mm:ss

```

Examples

- <skype:echo123>
- <skype:echo123?call>
- <skype:echo123?chat>

Notice that there is no "/" in skype: URI - `skype://echo123` does **not** work.

Release Notes

This section contains all release notes for Skype:

- [Skype 1.4 release notes](#)
- [Skype 1.3.0.42](#)
- [Skype 1.2.0.11](#)
- [Skype 1.1.0.61](#)
- [Skype 1.0.0.94](#)

Skype 1.4 Release Notes

Date: 2005-09-16

Changes and fixes:

- Support for application to application messaging
- Set profile properties
- Support for call forwarding
- Extended support to open client windows
- New user object properties (mood text, alias)
- Extended support for ringtones
- Support for Skype URI handler commands
- Support for contact focused notifications

Skype 1.3.0.42 release notes

Date: 2005-06-11

Changes and fixes:

- added: Protocol 5
 - Support for voicemails: VOICEMAIL, OPEN VOICEMAIL, ALTER VOICEMAIL, SEARCH VOICEMAILS
- Support for chat handling: CHAT CREATE, OPEN CHAT, ALTER CHAT, SEARCH *CHATS
- Support for authorizations: SEARCH USERSWAITINGMYAUTHORIZATION, SET USER, ISAUTHORIZED, ISBLOCKED, BUDDYSTATUS
- Support for deleting history: CLEAR CHATHISTORY, VOICEMAILHISTORY, CALLHISTORY
- Set ringing device: SET/GET RINGER
- Extended DTMF support: SET CALL DTMF
- Initiate filetransfer: OPEN FILETRANSFER
- Assign speeddial: USER SPEEDDIAL
- Change ringtones: GET/SET RINGTONE
- Change avatar: SET AVATAR
- Minimize Skype window: MINIMIZE
- bugfix: conference call bugs resolved

Skype 1.2.0.11 release notes

Date: 2005-03-04

Changes and fixes:

- added: Protocol 4
 - Support for conferencing: start a conference, add people to conference and being able to get list of conference call participants and notifications about these
- Possible to check SkypeOut balance
- Possible to call speeddial numbers
- Notifications about changing audiodrivers
- Notification about deleting IM history
- Changed language and country to return ISO list instead of countrynames (new behaviour: starting from protocol 4 language and country values will be prefixed by ISO codes ('GET USER echo123 COUNTRY' => 'USER echo123 COUNTRY ee Estonia'))
- Notification about shutting down Skype
- Support for Skypeln
- Registry key to disable one second timeout for debugging
- Possibility to add userhandle to OPEN ADDAFRIEND
- Support for command-id (#1 SET xxx)
- CALL FAILUREREASON 1 - documentation error, changed to say "Misc error"
- change: if CHATMESSAGE property is missing, command 'SET CHATMESSAGE id' gives the same error for both existing and nonexisting id
- change: PSTN_STATUS gives error string returned from gateway
- change: HASCALEQUIPMENT always returns TRUE
- change: Up/down via phone api autoexpand contactlist groups
- bugfix: "AUDIO IN" and "AUDIO OUT" commands do not read double byte driver names correctly
- bugfix: BTN_PRESSED E fails with error 71 invalid key
- bugfix: Muting microphone in UI not reflected in API
- bugfix: Conference to more than 4 participants causes "Range check" errors
- bugfix: IMHISTORYCHANGED doesn't work
- bugfix: SET MUTE ON returns always MUTE OFF
- bugfix: Cannot call SkypeOut contacts using speeddial
- bugfix: No response to empty CALL (should return ERROR 34 invalid user handle)s
- bugfix: Skype access control does not deny access to a device
- bugfix: No notification if the user changes audio device

Skype 1.1.0.61 release notes

Date: 2005-01-12

Changes and fixes:

- added: Protocol 3
- change: API - now allows one ongoing search per client only. Attempting to issue new search before receiving results of a previous one results in error 72.
- change: API allows now one ongoing search per client only
- change: `CHAT` and `CHATMESSAGE` properties
- bugfix: API showed previous user's calls and messages
- bugfix: Fixed confusing syntax if protocol 3 is used
- bugfix: `SEARCH MESSAGES` does not return `CHATMESSAGES` value anymore if protocol 2 is used
- bugfix: API displayed only first word of message or fullname
- bugfix: In access control list only one program's permission was remembered
- bugfix: Multichat message IDs were not returned
- bugfix: Problems with connecting for older applications
- bugfix: Fixed API exceptions if Skype is used on two Windows accounts simultaneously
- bugfix: On Windows98/ME some dll files were shown to use Skype instead of the respective application
- bugfix: Sometimes API didn't return '`BUDDYSTATUS 1`' messages

Skype 1.0.0.94 release notes

Date: 2004-10-21

Changes and fixes

Release of Skype public API.

Commands list

- [CALL](#)
- [OPEN VOICEMAIL](#)
- [ALTER VOICEMAIL](#)
- [CHAT CREATE](#)
- [CHATMESSAGE](#)
- [ALTER CHAT <id> SETTOPIC](#)
- [ALTER CHAT <id> ADDMEMBERS](#)
- [ALTER CHAT <id> LEAVE](#)
- [GET CHAT <id> CHATMESSAGES](#)
- [GET CHAT <id> CHATMESSAGES](#)
- [SET CHATMESSAGE <id> SEEN](#)
- [SET MESSAGE <id> SEEN](#)
- [MESSAGE](#)
- [SEARCH](#)
 - [SEARCH FRIENDS](#)
- [SEARCH USERS](#)
- [SEARCH CALLS](#)
 - [SEARCH ACTIVECALLS](#)



- [SEARCH MISSEDCALLS](#)
 - [SEARCH VOICEMAILS](#)
- [SEARCH MESSAGES](#)
 - [SEARCH MISSEDMESSAGES](#)
 - [SEARCH CHATS](#)
 - [SEARCH ACTIVECHATS](#)
- [SEARCH MISSEDCHATS](#)
- [SEARCH RECENTCHATS](#)
- [SEARCH BOOKMARKEDCHATS](#)
 - [SEARCH CHATMESSAGES](#)
 - [SEARCH MISSEDCHATMESSAGES](#)
 - [SEARCH USERSWAITINGMYAUTHORIZATION](#)
- [CLEAR CHATHISTORY](#)
- [CLEAR VOICEMAILHISTORY](#)
- [CLEAR CALLHISTORY](#)
- [FOCUS](#)
- [MINIMIZE](#)
- [OPEN ADDAFRIEND](#)
- [OPEN IM](#)
- [OPEN CHAT <id>](#)
- [OPEN FILETRANSFER](#)
- [OPEN VOICEMAIL](#)
- [PING](#)
- [GET USER <username> property](#)
- [SET USER <username> property value](#)
- [GET CALL <id> property](#)
- [GET CHAT <id> property](#)
- [GET CHATMESSAGE <id> property](#)
- [GET MESSAGE <id> property](#)
- [SET](#)
- [GET SKYPEVERSION](#)
- [GET PRIVILEGE user_privilege](#)
- [GET PROFILE profile_property](#)
- [GET/SET AUDIO_IN](#)
- [GET/SET AUDIO_OUT](#)
- [GET/SET RINGER](#)
- [GET/SET MUTE](#)
- [SET AVATAR](#)
- [GET/SET RINGTONE](#)
- [DELETE APPLICATION](#)
- [CREATE APPLICATION](#)
- [DELETE APPLICATION](#)
- [APPLICATION WRITE](#)
- [APPLICATION DATAGRAM](#)
- [APPLICATION READ](#)
- [CALL FORWARDING](#)