GO-GLOBAL

Client Process Manager API

Version 4.8.2

COPYRIGHT AND TRADEMARK NOTICE

Copyright © 1997-2015 GraphOn Corporation. All Rights Reserved.

This document, as well as the software described in it, is a proprietary product of GraphOn, protected by the copyright laws of the United States and international copyright treaties. Any reproduction of this publication in whole or in part is strictly prohibited without the written consent of GraphOn. Except as otherwise expressly provided, GraphOn grants no express or implied right under any GraphOn patents, copyrights, trademarks or other intellectual property rights. Information in this document is subject to change without notice.

GraphOn, the GraphOn logo, and GO-Global and the GO logo are trademarks or registered trademarks of GraphOn Corporation in the US and other countries. Microsoft, Windows, Windows NT, Internet Explorer, and Terminal Server are trademarks of Microsoft Corporation in the United States and/or other countries. Linux is a registered trademark of Linus Torvalds. UNIX is a registered trademark of The Open Group. Red Hat is a trademark or registered trademark of Red Hat, Inc. in the United States and other countries. Adobe, Acrobat, AIR, Flash, and Reader are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries. Firefox is a registered trademark of the Mozilla Foundation. Mac, Mac OS, and Safari are trademarks of Apple Inc., registered in the U.S. and other countries.

Portions copyright © 1998-2000 The OpenSSL Project. All rights reserved. This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (www.openssl.org). Portions copyright © 1995-1998 Eric Young (eay@cryptsoft.com). All rights reserved. This product includes software written by Eric Young (eay@cryptsoft.com). This product includes software written by Tim Hudson (tjh@cryptsoft.com).

All other brand and product names are trademarks of their respective companies or organizations.

Printed in the United States of America.

ABSTRACT

This document describes the GO-Global Client Process Manager API, a set of functions that applications running in GO-Global sessions on a server can call to launch and manage native processes on the computer that is running the GO-Global client.

The Client Process Manager API is supported on the Windows client only.

API SPECIFICATION

unsigned long ProcessId;
Defines a process identifier that can be used to uniquely identify a client process

ProcessId createClientProcess(unsigned short* filename, unsigned short* commandLine);
@param filename, name of the executable or document
@param commandLine, the command line passed to the new process when filename is an executable
@return ProcessId, the identifier for the new process, or zero if an error occurred.

Creates a new process on the client machine using the executable associated with the specified filename. If *filename* represents a document type, the new process will be created using the executable the shell uses to open the file. If *filename* specifies an executable, then the client path is searched for a matching file. If the complete path to the file is not specified in filename then the client will try to locate it.

The *commandLine* argument is only used if *filename* is an executable. If *filename* is a document, it will be passed on to the command line to the associated executable and the *commandLine* argument will be ignored.

Both *filename* and *commandLine* are 16-bit null-terminated character arrays. Single and multi-byte character sets are not supported.

If *filename* specifies the complete path to a document that resides on the server then a copy of the document will be created on the client and that file will be used to create the new process. If *filename* contains the complete path to a server executable then the path is removed and the client path is searched for the executable.

Path information in both *filename* and *commandLine* is automatically mapped to the client's file system

boolean destroyClientProcess(ProcessId pid);
@param pid, the identifier of the process to destroy
@return boolean, returns true if successful, false if the timeout was reached or an error occurred.

Destroys the specified process. This function should not be used in conjunction with *getClientProcessExitCode*.

boolean waitForClientProcessCompletion(ProcessId pid, unsigned long timeout);
@param pid, the identifier of the process to wait for.
@param timeout, the amount of time to wait for completion, in milliseconds
@return boolean, returns true if successful, false if the timeout was reached or an error occurred.

Waits for the specified process to complete execution, or until the specified timeout has elapsed.

boolean getClientProcessExitCode(ProcessId pid, unsigned long* exitCode);
@param pid, the identifier of the process being queried.
@param exitCode, pointer to an integer value to receive the exit code.
@return boolean, returns true if successful, false if the timeout was reached or an error occurred.

If successful, fills the integer pointed to by *exitCode* with the value that the specified client process returned when exiting. The function will fail if the process is still running, or if the process identifier is invalid or has already been closed.

To wait until the process has completed execution, call *waitForClientProcessCompletion* before calling this function.

boolean closeClientProcess(ProcessId pid);
@param pid, the identifier of the process being closed
@return boolean, returns true if successful, false if the timeout was reached or an error occurred.

Frees all resources associated with the client process, but does not terminate the process. Must be called once for each non-zero ProcessId returned by the createClientProcess function. The process identifier becomes invalid after the successful completion of this function.

unsigned long getLastError();
@return unsigned long, the last error code set using *setLastError*.

Returns the last error code set for the current thread. Call this function to retrieve extended error information when another function returns a failure condition.

unsigned long setLastError(unsigned long error);
@param error, the new error code for the current thread.
@return unsigned long, the last error code set using setLastError.

Sets the error code for the current thread. Useful for clearing the thread-specific error value before calling a function that may subsequently set it.