SafeGuard LM

Table of Contents

Introduction:

The *SafeGuard LM* “C” API provides the interface for your C/C++, Objective C, Java, Python, VB.NET and PHP 5 applications to utilize license management in your software applications. Whether it is for floating licenses or node-locked licenses or both. The API also includes utility functions to help manage your license deployment and installation along with some other helpful utilities.

Refer to the SystemRequirements.pdf for information on Compilers, linkers and supported Operating Systems. See the DeveloperGuide.pdf for file layout and required libraries.

In brief, here are examples of utilizing SafeGuard LM from your favorite language.

[C/C++]

Add the following to your .c/.cpp and include the appropriate psprolib library to your project

```
#include “psprolib.h”
```

[Objective C]

Add the following to your .m and include SafeGuardLM.framework to your xcode project

```
#import "PSProlibProxy.h"
```

[Java]

Import the following when writing your application.

```
import com.persistentsecurity.safeguardlm.*;
```

[Python]

```
Python 2.7.3 (v2.7.3:70274d53c1dd, Apr  9 2012, 20:52:43)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import PSProlibProxy
>>>
```

[VB.NET]

Add psprolib.vb to your VB project.

[PHP 5]

Add the following to your php script

NOTE: The PHP 5.6.17 API is only available on OS X and Linux.

```
include ("psprolib.php");
```

Before running php from a terminal window you must first copy the PHP version of psprolib.so to one of the following locations after installing php on your system.

```
copy pspro_xxxNN/php/psprolib.so to /usr/local/lib/php/extensions/no-debug-non-zts-20131226
```

(Note: On Raspbian the location is /usr/lib/php5/20100525+lfs)

[MetaTrader 5]

In your MetaTrader 5 script or Indicator, you will need to include the psprolib.mqh. This header file can be placed in your project folder or in the MT5\Include folder within the MetaTrader installation folder.

```
#include “psprolib.mqh”
```

MetaTrader “strings” are UNICODE. SafeGuard LM “strings” are UTF-8 null terminated char arrays with possible imbedded ASCII text. When passing a MT string to SafeGuard, it will have to be converted to a char array first. When a SafeGuard LM function returns a string, it will need to be converted back to a UNICODE string before being used within MT. Here are the two MT functions you should use to perform this.

```
string CharArrayToString(ansi_srting,0,WHOLE_ATTRRAY,CP_UTF8);

char ansi_string[256];
StringToCharArray("unicode-string",ansi_string,0,WHOLE_ARRAY,CP_UTF8);
```

[FORTRAN]

The fortran API is built into the standard C/C++ psprolib libraries for all platforms. The calling sequence is very similar to C with the major difference that fortran passes integer arguments by reference (address) whereas C passes integer arguments by value. So where you might pass a &returnvalue argument to a C Function, in fortran you would just pass the integer variable as returnvalue. There is also a psprolib.inc in the header directory. Each fortran program or function will need to include this header file to get you going. On Windows, only the DLL version of psprolib can be used.

Global license implementation functions:

These API's are used by both floating and node-locked license implementations. For example, **sgSetAttrInt()** with **SG_ATTR_LICENSETYPE** must be called before using a license implementation other than node-locked external licenses. So, you would need to call this function with the appropriate license type before checking out a floating license.

The attribute setters and getters can be used to configure or retrieve several other licensing attributes that you may wish to control during execution of your application.

sgGetLastErrorString()

Retrieve the last error number and error string.

```
[C/C++]
int retVal;
char message[512];
retVal = sgGetLastErrorString(message);
```

```
[Objective C]
NSInteger retVal
NSString *message;
retVal = [PSProlibProxy sgGetLastErrorString:self  getMessage:&message];
```

```
[Java]
int retVal.days;
byte message[] = new byte[512];
retVal = PSProlibProxy.sgGetLastErrorString(message);
```

```
[Python]
retVal, message = PSProlibProxy.sgGetLastErrorString()
```

```
[Visual Basic .NET]
Dim retVal As Integer
Dim message As New StringBuilder(512)
retVal =  PSProlibProxy.sgGetLastErrorString(message)
```

```
[PHP 5]
$message = str_repeat(' ',512);
$retVal = sgGetLastErrorString($message);
```

```
[MT5]
char message[512];
int retVal;
retVal = sgGetLastErrorString(message);
```

```
[FORTRAN]
character*512 message
integer retVal
retVal = sgGetLastErrorString(message)
```

Parameters

message [out]

The last error string returned in the currently set language setting

Description

Returns the last error code and message. The message string will be localized in accordance with the **SG_ATTR_LANGUAGE** setting. The default language setting is **SG_LANGUAGE_ENG**.

Return codes

Possible return codes are:

SG_SUCCESS
SG_AUTHORIZED
SG_TARGET_FILE_NOT_FOUND
SG_NOT_AUTHORIZED_FOR_COMPONENT
SG_NOT_AUTHORIZED_FOR_VERSION
SG_COMPONENT_EXPIRED
SG_CANT_CREATE_TARGET_FILE
SG_CANT_GET_ID
SG_INVALID_CERTIFICATE
SG_INVALID_TARGET_INFORMATION
SG_NO_MORE_COMPONENTS_IN_FILE
SG_NO_COMPONENT_MATCHING_ID
SG_COMPONENT_NAME_TOO_LONG
SG_CANT_WRITE_TO_TARGET_FILE
SG_CANT_READ_TARGET_FILE
SG_INVALID_ID_LENGTH
SG_CANT_CREATE_TEMPORARY_FILE
SG_CANT_OPEN_TEMPORARY_FILE
SG_CANT_DELETE_TARGET_FILE
SG_CANT_RENAME_TARGET_FILE
SG_CANT_READ_TEXT_FILE
SG_CANT_WRITE_TEXT_FILE
SG_CANT_DELETE_TEXT_FILE
SG_CANT_RENAME_TEXT_FILE
SG_TEXT_FILE_NOT_FOUND
SG_NO_COMPONENT_FOUND_IN_FILE
SG_NO_FREE_SLOTS
SG_INVALID_TIMEOUT
SG_VERSION_LENGTH_TOO_LONG
SG_IDTYPE_NOT_SUPPORTED
SG_BAD_LICENSETYPE
SG_ATTRTYPE_NOT_SUPPORTED
SG_ATTRTYPE_INVALID
SG_CANT_CREATE_LOGFILE
SG_LOGFILE_NOT_SET
SG_INVALID_OPTION_FOR_LICENSETYPE
SG_NO_MORE_CLIENT_COMPONENTS
SG_NO_MORE_SERVER_COMPONENTS
SG_INVALID_LICENSE_KEY
SG_VERSION_NOT_NUMERIC
SG_NOT_A_CALLABLE_OBJECT
SG_INVALID_LICENSE_COUNT
SG_SERVERNORESPOND
SG_BADCOMMUNICATION
SG_INVALIDHOST
SG_BADCONNECT
SG_BADREAD
SG_BADWRITE
SG_MAXSERVERCONNECTIONS
SG_LOSTCONNECT

SG_ALL_LICENSES_IN_USE
SG_NO_SUCH_COMPONENT
SG_NOT_LICENSED_FOR_VERSION
SG_NOT_CHECKED_OUT
SG_TIMEDIFF_TOO_LARGE
SG_COMPONENT_NOT_AVAILABLE
SG_COMPONENT_ALREADY_ISSUED
SG_BAD_PASSWORD
SG_CANT_QUERY_SQLDB
SG_CANT_CONNECT_TO_SQLDB
SG_WRONG_SERVER
SG_REJECTED
SG_NOT_A_CALLABLE_OBJECT
SG_SYSTEM_CLOCK_SET_BACK
SG_UNKNOWN_ERROR

sgSetAttrStr()

Sets any number of string attributes for the currently running process.

```
[C/C++]
int retVal;
retVal = sgSetAttrStr(SG_ATTR_LICENSE_FILE, “29750@serverhost”);
```

```
[Objective C]
NSInteger retVal;
retVal= [PSProlibProxy sgSetAttrStr:self forAttrType:SG_ATTR_LICENSE_FILE
                                                  forValue:@"29750@serverhost"];
```

```
[Java]
int retVal;
retVal = PSProlibProxy.sgSetAttrStr(SG_ATTRTYPE.SG_ATTR_LICENSE_FILE,”29750@serverhost”);
```

```
[Python]
retVal = PSProlibProxy.sgSetAttrStr(PSProlibProxy.SG_ATTR_LICENSE_FILE,”29750@serverhost”)
```

```
[Visual Basic .NET]
Dim retVal As Integer
retVal = PSProlibProxy.sgSetAttrStr(psprolib.SG_ATTRTYPE.SG_ATTR_LICENSE_FILE,”29750@serverhost”)
```

```
[PHP 5]
$retVal = sgSetAttrStr(SG_ATTR_LICENSE_FILE,'29750@serverhost')
```

```
[MT 5]
char attrval[128];
int retVal;
StringToCharArray("29750@serverhost", attrval, 0, WHOLE_ARRAY, CP_UTF8);
retVal = sgSetAttrStr(SG_ATTR_LICENSE_FILE, attrval);
```

```
[FORTRAN]
integer retVal
retVal = sgSetAttrStr(SG_ATTR_LICENSE_FILE, '29750@serverhost')
```

Parameters

attrtype [in]

SG_ATTR_LOGFILE_PATH	**SG_ATTR_VENDOR_ROUTINE**
SG_ATTR_VENDOR_LIBRARY	**SG_ATTR_LICENSE_FILE**

attrval [in]

An appropriate value for the possible attribute types

Description

The following attributes can be modified to control aspects of the running application.

SG_ATTR_LOGFILE_PATH

This can be used to set the log filename for recording **sg()** function calls. If the file does not exist, it will be created. If it already exists, it will be appended to. Unless the full pathname is given, it will be created in the current working directory. If called more than once with a different output filename, the log file with be switched to the new filename.

SG_ATTR_VENDOR_ROUTINE
SG_ATTR_VENDOR_LIBRARY

These are used if you have programmed your own vendor defined host ID. In other words, you wish to use an IdType other than the ones provided. This may be the case if you have your own hardware key (dongle), or you would like to support some other IdType which can literally be anything. For example, lets say you wanted to license your software by timezone. You would code a function that retrieves the current timezone and use these attributes to tell SafeGuard LM to use your ID retrieval function. Then when you generate your licenses, you specify the “vendor” IdType and the correct timezone value you wish to license, and your software will only work in that timezone when requesting a license. See *examples/c/vendorid-dynamic.c* as an example

These set the vendor defined function that returns a vendor defined ID. The **SG_ATTR_VENDOR_ROUTINE** is the name of the function. The **SG_ATTR_VENDOR_LIBRARY** is the dynamic library that contains the function. The library name should be the complete filename including the extension. If the library value is blank, then only the main application is searched for the function. This is used when the get ID function is linked statically into the main application. This call is not valid when **SG_LICENSETYPE_FLOATING** is set.

SG_ATTR_LICENSE_FILE

For floating license applications, this allows you to specify the license server to connect to. The value can be either a license file pathname or in the format of PORT@HOST. If a license file pathname is specified, then the port and server name from the license file will be used. The license file will need to be accessible from the client machine. If the PORT@HOST format is specified, then that will be used to contact the license server. By default, SafeGuard LM uses 29750@localhost If PORT@HOST is used, the port number must match the port number in the floating license file. Keep in mind that the user can override this setting unless you disable the use of environment the variable **SG_LICENSE_FILE** by specifying **SG_ATTR_DISABLE_ENV**.

Return codes

On success, **SG_SUCCESS** is returned

On error, possible return codes are:

SG_ATTRTYPE_NOT_SUPPORTED
SG_CANT_CREATE_LOGFILE

sgSetAttrInt()

Sets any number of integer attributes for the currently running process.

```
[C/C++]
int retVal;
retVal = sgSetAttrInt(SG_ATTR_LICENSETYPE, SG_LICENSETYPE_FLOATING);
```

```
[Objective C]
NSInteger retVal;
retVal = [PSProlibProxy sgSetAttrInt:self forAttrType:SG_ATTR_LANGUAGE
forValue:SG_LANGUAGE_SPA];
```

```
[Java]
int retVal;
retVal = PSProlibProxy.sgSetAttrInt(SG_ATTRTYPE,SG_ATTR_TCP_TIMEOUT, 4);
```

```
[Python]
retVal =
PSProlibProxy.sgSetAttrInt(PSProlibProxy.SG_ATTR_LICENSETYPE,PSProlibProxy.SG_LICENSETYPE_FLO
ATING)
```

```
[Visual Basic .NET]
Dim retVal As Integer
retVal = PSProlibProxy.sgSetAttrInt(psprolib.SG_ATTRTYPE.SG_ATTR_DISALLOW_IDTYPE,
                                    psprolib.SG_IDTYPE.SG_IDTYPE_ANY)
```

```
[PHP 5]
$retVal = sgSetAttrInt(SG_ATTR_LANGUAGE,SG_LANGUAGE_FRA);
```

```
[MT5]
int retVal;
retVal = sgSetAttrInt(SG_ATTR_LICENSETYPE,SG_LICENSETYPE_FLOATING);
```

```
[FORTRAN]
integer retVal
retVal = sgSetAttrInt(SG_ATTR_LICENSETYPE, SG_LICENSETYPE_FLOATING)
```

Parameters

attrtype [in]

SG_ATTR_DISALLOW_IDTYPE	**SG_ATTR_MAXIMUM_TIMEDIFF**
SG_ATTR_DISALLOW_ALL_IDTYPE	**SG_ATTR_ALLOW_IDTYPE**
SG_ATTR_ALLOW_ALL_IDTYPE	**SG_ATTR_LICENSETYPE**
SG_ATTR_LANGUAGE	**SG_ATTR_TCP_TIMEOUT**
SG_ATTR_RETRY_INTERVAL	**SG_ATTR_RETRY_COUNT**
SG_ATTR_DISABLE_ALARM	**SG_ATTR_DISABLE_ENV**
SG_ATTR_DATECHECK	**SG_ATTR_DISABLE_LANG**

attrval [in]

One of the following attributes or an integer value when specifying such things as TCP/IP timeout

SG_IDTYPE_DEFAULT	**SG_IDTYPE_IPHONE**	**SG_IDTYPE_ANDROID**
SG_IDTYPE_BLACKBERRY	**SG_IDTYPE_ETHERNET**	**SG_IDTYPE_HOSTNAME**

SG_IDTYPE_IPADDRESS	**SG_IDTYPE_USERNAME**	**SG_IDTYPE_COUNTRY**
SG_IDTYPE_INET6	**SG_IDTYPE_VSN**	**SG_IDTYPE_VENDOR**
SG_IDTYPE_ANY	**SG_IDTYPE_OS**	**SG_IDTYPE_HOSTID**
SG_IDTYPE_MACINTOSH	**SG_IDTYPE_METERED**	**SG_IDTYPE_COMPOSITE**
SG_IDTYPE_METERED	**SG_LANGUAGE_DEFAULT**	**SG_LANGUAGE_DEU**
SG_LANGUAGE_ENG	**SG_LANGUAGE_SPA**	**SG_LANGUAGE_FRA**
SG_LANGUAGE_ITA	**SG_LANGUAGE_POR**	**SG_LANGUAGE_TUR**
SG_LANGUAGE_KOR	**SG_LANGUAGE_GER**	**SG_LANGUAGE_DET**
SG_LANGUAGE_RUM	**SG_LANGUAGE_UKR**	**SG_LANGUAGE_JPN**
SG_LANGUAGE_SRP	**SG_LANGUAGE_SWE**	**SG_LANGUAGE_DAN**
SG_LANGUAGE_SLO	**SG_LANGUAGE_SLV**	
SG_LICENSETYPE_DEFAULT		**SG_LICENSETYPE_EXTERNAL**
SG_LICENSETYPE_INTERNAL		**SG_LICENSETYPE_FLOATING**
SG_DATECHECK_STANDARD		**SG_DATECHECK_EXTENDED**

Description

The following attributes can be modified to control aspects of the running application.

SG_ATTR_DISALLOW_IDTYPE

This can be used to disable certain IdTypes from being used. Lets say you don't want your application to ever be able to use the “any” host ID type. This is used for security reasons. **We STRONGLY suggest you never use SG_ATTR_DISALLOW_ALL_IDTYPE, as the software uses certain id types like SG_IDTYPE_OS for certain internal functions and they will fail if you are not very careful.**

SG_ATTR_DISALLOW_ALL_IDTYPE
SG_ATTR_ALLOW_IDTYPE

This works the same as **SG_ATTR_DISALLOW_IDTYPE** only it disables all IdTypes (pass SG_ATTR_NULL as the second argument when disabling all IdTypes). In essence, disabling every license. Use this if you would only want to allow one or a couple of IdTypes by first using this, followed by **SG_ATTR_ALLOW_IDTYPE**.

SG_ATTR_ALLOW_ALL_IDTYPE

This works the same manner as SG_ATTR_DISALLOW_IDTYPE only it enables ALL IdTypes. In essence, allowing all IdTypes to be able to be used. By default, all IdTypes are enabled for each platform architecture. (pass SG_ATTR_NULL as the second argument when enabling all IdTypes)

SG_ATTR_LICENSETYPE

This sets the license type for the application to either node-locked external (the default), node-locked internal or floating licenses. This should be called early on in your application before any license checks are performed.

SG_ATTR_LANGUAGE

This sets the language type for the application. Subsequent calls to **sgGetLastErrorString()** will return the messages in the selected language.

SG_ATTR_TCP_TIMEOUT

This overrides the default 2 second TCP/IP connection timeout when the license type is set to floating.

2 seconds is a reasonable timeout, but if you wish to change it to say 4 seconds, this is how you do it.

SG_ATTR_RETRY_INTERVAL

For floating license types this can be used to override the default 2 minute retry interval for your application to attempt to reconnect with the license server if it were to lose the connection. Unless you have disabled the automatic timer using **SG_ATTR_DISABLE_ALARM** (OS X, Linux), your application will automatically attempt to reconnect with the license server at this minute interval.

SG_ATTR_RETRY_COUNT

For floating license types this can be used to override the default 5 retry count for your application when it attempt to reconnect with the license server if it were to lose the connection. Unless you have disabled the automatic timer using **SG_ATTR_DISABLE_ALARM** (OS X, Linux), your application will automatically attempt to reconnect with the license server 5 times, at which time it will automatically terminate.

SG_ATTR_DISABLE_ALARM

For floating license applications that cannot tolerate SIGALRM (Linux, OS X), or don't want the application to check for a valid connection to the license server (all platforms), you can disable this check as long as it is done before the application makes its first connection to the license server. Be aware, if you disable this check, licenses can be stolen from the server by restarting the server after a license has been checked out and all licenses would be returned to the license pool unless you perform a **sgTimer()** call yourself in your application on a regular interval. The value must be non-zero to disable this check.

SG_ATTR_DISABLE_ENV

For floating licenses, this will disallow the ability for users to override the license path by setting the environment variable **SG_LICENSE_FILE**. The value must be non-zero to disable this check.

SG_ATTR_DISABLE_LANG

This is used to disable the ability for the user to override the default or current language setting. Users can change the default or current language settings by setting an environment variable SG_LANGUAGE to any of the supported ISO 639-2 language codes. However, you can disable this feature by passing a non-zero value as the parameter.

SG_ATTR_MAXIMUM_TIMEDIFF

For floating licenses, this can be used to change the default 36 hour time difference between the client machine and the license server. If it greater than the default or specified hour difference, the license functions will not work. All timezones in the world are taken into account. So, if you are on a ship crossing timezones and your computers update your current timezone, as long as your license server is not more than the default or specified number of hours from where you are in the world given the current timezone, you are ok to check out licenses.

SG_ATTR_DATECHECK

For node-locked licenses, this can be use to perform extended date checking to see whether the user has turned back their system clock, or are running programs such as “Date Cracker 2000” to circumvent the timeout of the component. **SG_DATECHECK_STANDARD** is the default, **SG_DATECHECK_EXTENDED** will thwart the above methods of circumventing the timeout. This is currently for Windows only.

On success, **SG_SUCCESS** is returned.

On error, possible return codes are:

SG_ATTRTYPE_NOT_SUPPORTED
SG_IDTYPE_NOT_SUPPORTED

sgGetAttrStr()

Gets any number of string attributes for the currently running process.

[C/C++]
int retVal;
char attrval[256];
retVal = sgGetAttrStr(SG_ATTR_LICENSE_FILE, attrval);

[Objective C]
NSInteger retVal;
NSString *attrval;
retVal = [PSProlibProxy sgGetAttrStr:self forAttrType:SG_ATTR_EXE_PATH getValue:&attrval];

[Java]
int retVal;
byte message[] = new byte[512];
retVal = PSProlibProxy.sgGetAttrStr(SG_ATTRTYPE.SG_ATTR_ETHERNET_ID,attrval);

[Python] **retVal, attrval = PSProlibProxy.sgGetAttrStr(PSProlibProxy.SG_ATTR_ARCHITECTURE)**
[Visual Basic .NET] **Dim retVal As Integer** **Dim attrval As StringBuilder[256]** **retVal = PSProlibProxy.sgGetAttrStr(psprolib.SG_ATTRTYPE.SG_ATTR_RELEASE,attrval)**
[PHP 5] **$attrval = str_repeat(' ',256);** **$retVal = sgGetAttrStr(SG_ATTR_LICENSE_FILE, $attrval);**

[MT5]
int retVal;
char attrval[256];
retVal = sgGetAttrStr(SG_ATTR_USERNAME_ID,attrval);

[FORTRAN] **integer retVal** **character*256 attrval** **retVal = sgGetAttrStr(SG_ATTR_LICENSE_FILE, attrval)**

Parameters

attrtype [in]

SG_ATTR_DEFAULT_ID	**SG_ATTR_IPHONE_ID**	**SG_ATTR_ANDROID_ID**
SG_ATTR_BLACKBERRY_ID	**SG_ATTR_ETHERNET_ID**	**SG_ATTR_HOSTNAME_ID**
SG_ATTR_IPADDRESS_ID	**SG_ATTR_USERNAME_ID**	**SG_ATTR_COUNTRY_ID**
SG_ATTR_INET6_ID	**SG_ATTR_VSN_ID**	**SG_ATTR_DEFAULT_ID**
SG_ATTR_IPHONE_ID	**SG_ATTR_ANDROID_ID**	**SG_ATTR_BLACKBERRY_ID**
SG_ATTR_ETHERNET_ID	**SG_ATTR_HOSTNAME_ID**	**SG_ATTR_IPADDRESS_ID**
SG_ATTR_USERNAME_ID	**SG_ATTR_COUNTRY_ID**	**SG_ATTR_INET6_ID**
SG_ATTR_VSN_ID	**SG_ATTR_VENDOR_ID**	**SG_ATTR_ANY_ID**
SG_ATTR_OS_ID	**SG_ATTR_HOSTID_ID**	**SG_ATTR_MACINTOSH_ID**
SG_ATTR_METERED_ID	**SG_IDTYPE_COMPOSITE_ID**	**SG_ATTR_LOGFILE_PATH**

SG_ATTR_LICENSE_FILE	**SG_ATTR_RELEASE**	**SG_ATTR_ARCHITECTURE**
SG_ATTR_EXE_PATH	**SG_ATTR_VENDOR_ROUTINE**	
SG_ATTR_VENDOR_LIBRARY		

attrval [out]

The value of the given attribute

Description

The following attributes can be retrieved from the running application.

SG_ATTR_DEFAULT_ID
SG_ATTR_IPHONE_ID
SG_ATTR_ANDROID_ID
SG_ATTR_BLACKBERRY_ID
SG_ATTR_ETHERNET_ID
SG_ATTR_HOSTNAME_ID
SG_ATTR_IPADDRESS_ID
SG_ATTR_USERNAME_ID
SG_ATTR_COUNTRY_ID
SG_ATTR_INET6_ID
SG_ATTR_VSN_ID
SG_ATTR_VENDOR_ID
SG_ATTR_ANY_ID
SG_ATTR_OS_ID
SG_ATTR_HOSTID_ID
SG_ATTR_MACINTOSH_ID
SG_ATTR_METERED_ID
SG_IDTYPE_COMPOSITE_ID

These get one of the ID types for the current machine if supported. Not all ID types are available on each platform.

SG_ATTR_LOGFILE_PATH

This returns the current log file pathname; if set.

SG_ATTR_VENDOR_ROUTINE

Returns the name of the vendor defined hostid function name if defined. This is used only when a developer has created their own vendor defined hostid.

SG_ATTR_VENDOR_LIBRARY

Returns the name of the vendor defined hostid function name if defined. This is used only when a developer has created their own vendor defined hostid.

SG_ATTR_LICENSE_FILE

Returns the current license file path or PORT@HOST for floating license types.

SG_ATTR_RELEASE

returns the current release number of SafeGuard LM.

SG_ATTR_ARCHITECTURE

Returns the SafeGuard LM platform architecture. Values can be, mac32, mac64, win32, win64, lin32, lin64.

SG_ATTR_EXE_PATH

Returns the pathname to the currently running application.

On success, **SG_SUCCESS** is returned.

On error, possible return codes are:

SG_ATTRTYPE_NOT_SUPPORTED
SG_LOGFILE_NOT_SET

sgGetAttrInt()

Gets any number of numeric attributes for the currently running process.

[C/C++]
```
int retVal,attrval;
retVal = sgGetAttrInt(SG_ATTR_TCP_TIMEOUT, &attrval);
```

[Objective C]
```
NSInteger retVal, attrval;
retVal = [PSProlibProxy sgGetAttrInt:self forAttrType:SG_ATTR_TCP_TIMEOUT getValue:&attrval];
```

[Java]
```
int retVal, attrval;
SWIGTYPE_p_int attrvalPtr;
retVal = PSProlibProxy.sgGetAttrInt(SG_ATTRTYPE.SG_ATTR_TCP_TIMEOUT,attrvalPtr);
attrval = PSProlibProxy.intPtr_value(attrvalPtr);
PSProlibProxy.delete_intPtr(attrvalPtr);
```

[Python]
```
retVal, attrval = PSProlibProxy.sgGetAttrInt(PSProlibProxy.SG_ATTR_TCP_TIMEOUT)
```

[Visual Basic .NET]
```
Dim retVal As Integer
Dim attrval As Integer
retVal = PSProlibProxy.sgGetAttrInt(psprolib.SG_ATTRTYPE.SG_ATTR_TCP_TIMEOUT,attrval)
```

[PHP 5]
```
$attrPtr = new_intp();
sgGetAttrInt(SG_ATTR_TCP_TIMEOUT,$attrPtr);
$attrval =  intp_value($attrPtr);
delete_intp($attrPtr);
```

[MT5]
```
int retVal, attrval;
retVal = sgGetAttrInt(SG_ATTR_TCP_TIMEOUT,attrval);
```

[FORTRAN]
```
integer retVal,attrval
retVal = sgGetAttrInt(SG_ATTR_TCP_TIMEOUT, attrval)
```

Parameters

attrtype [in]

SG_ATTR_LICENSETYPE	**SG_ATTR_LANGUAGE**
SG_ATTR_TCP_TIMEOUT	**SG_ATTR_RETRY_INTERVAL**
SG_ATTR_RETRY_COUNT	**SG_ATTR_DISABLE_ALARM**
SG_ATTR_DISABLE_ENV	**SG_ATTR_MAXIMUM_TIMEDIFF**
SG_ATTR_DATECHECK	

attrval [out]

The value of the given attribute

Description

The following attributes can be retrieved from the running application.

SG_ATTR_LICENSETYPE

This returns the current license type. The default license type is SG_LICENSETYPE_EXTERNAL, which is the external text file license.

SG_ATTR_LANGUAGE

Returns the current language setting.

SG_ATTR_TCP_TIMEOUT

Returns the current TCP/IP timeout for floating license requests from the client to the license server.

SG_ATTR_RETRY_INTERVAL

For floating license types this can be used to retrieve the current retry interval in minutes for your application to attempt to reconnect with the license server if it were to lose the connection.

SG_ATTR_RETRY_COUNT

For floating license types this can be used to retrieve the current retry count for your application when it attempt to reconnect with the license server if it were to lose the connection.

SG_ATTR_DISABLE_ALARM

This returns the state of the alarm used for floating licenses. The alarm is used to make sure a connection is maintained for the life of the application. A 0 indicates the alarm has not been disabled, a 1 indicates it has been disabled.

SG_ATTR_DISABLE_ENV

Indicates whether the user can override the default or currently set license path for floating licenses. If it has been disabled, the user cannot override the this with the environment variable SG_LICENSE_FILE. A value of 0 indicates it has not been disabled, a value of 1 indicates it has been disabled.

SG_ATTR_DISABLE_LANG

Check whether the application has disable the ability for the user to override the language setting. A value of 0 indicates it has not been overridden, a value of 1 indicates it has.

SG_ATTR_MAXIMUM_TIMEDIFF

Returns the maximum allowable time difference between the client application and the license server. The default is 36 hours. The range can be between 1 and 336 (two weeks) hours.

SG_ATTR_DATECHECK

Returns the current setting for standard or extended date checking.

On success, **SG_SUCCESS** is returned.

On error, possible return codes are:

SG_ATTRTYPE_NOT_SUPPORTED

Node locked license implementation functions:

These are your main node-locked license functions. They are used to check whether your application is authorized to run, check when a particular license will expire and so on. These should be used in conjunction with the attribute setters and getters.

sgAuthorized()

Determines whether the application is licensed for the specified component.

[C/C++]
```
int retVal;
retVal = sgAuthorized(“@mylicense.lic”,”tree”,”4.0”);
```

[Objective C]
```
NSInteger retVal;
retVal = [PSProlibProxy sgAuthorized:self forTarget:@”@mylicense.lic” forComponent:@”tree”
                                                  forVersion:@”4.0”];
```

[Java]
```
int retVal;
retVal = PSProlibProxy.sgAuthorized(“$HOME/myapp/mylicense.lic”,”tree”,”4.0”);
```

[Python]
```
retVal = PSProlibProxy.sgAuthorized(“$HOME/myapp/mylicense.lic”,”tree”,”4.0”)
```

[Visual Basic .NET]
```
Dim retVal As Integer
retVal =  PSProlibProxy.sgAuthorized(“@mylicense.lic”,”tree”,”4.0”)
```

[PHP 5]
```
$retVal = sgAuthorized('$HOME/Sites/mylicense.lic','tree','4.0');
```

[MT5]
```
char target[256],component[32],version[10];
int retVal;
StringToCharArray("%HOME%\\MT5\\trademaster.lic",target,0,WHOLE_ARRAY,CP_UTF8);
StringToCharArray("trademaster",component,0,WHOLE_ARRAY,CP_UTF8);
StringToCharArray("5.0",version,0,WHOLE_ARRAY,CP_UTF8);
retVal = sgAuthorized(target,component,version);
```

[FORTRAN]
```
integer retVal
retVal = sgAuthorized('@mylicense.lic','tree','4.0')
```

Parameters

target [in]

The license file path for **SG_LICENSETYPE_EXTERNAL** licenses. The application path for **SG_LICENSETYPE_INTERNAL** licenses. For **SG_LICENSETYPE_EXTERNAL** licenses, you can specify “@mylicense.lic” for example. The “@” tells the software the license file resides next to the application executable. This makes it easy to locate a license file without having to construct a pathname. You may also use environment variables and relative pathnames here too for Windows, UNIX and Linux. **NOTE:** Do not use the @ symbol for Java or Python, as the software will look for the license file next to the Java or Python binary, e.g, /usr/bin/java, not where your java code is located.

component [in]

The component name.

version [in]

The version number of the component

Description

Determines whether the application is licensed for the specified component. This call is not valid when **SG_LICENSETYPE_FLOATING** is set.

Return codes

On success, **SG_AUTHORIZED** is returned.

On error, possible return codes are:

SG_NOT_AUTHORIZED_FOR_COMPONENT
SG_TARGET_FILE_NOT_FOUND
SG_CANT_READ_TARGET_FILE
SG_COMPONENT_NAME_TOO_LONG
SG_VERSION_LENGTH_TOO_LONG
SG_COMPONENT_EXPIRED
SG_NOT_AUTHORIZED_FOR_VERSION
SG_NO_COMPONENT_MATCHING_ID
SG_INVALID_OPTION_FOR_LICENSETYPE
SG_SYSTEM_CLOCK_SET_BACK

sgExpireDays()

Retrieve the number of days until the specified component expires.

```
[C/C++]
int retVal,days;
retVal = sgExpireDays(“@mylicense.lic”,”tree”,”4.0”,&days);
```

```
[Objective C]
NSInteger retVal,days;
retVal = [PSProlibProxy sgExpireDays:self forTarget:@”@mylicense.lic” forComponent:@”tree”
                                                  forVersion:@”4.0” getDays:&days];
```

```
[Java]
int retVal, days;
SWIGTYPE_p_int daysPtr;
retVal = PSProlibProxy.sgExpireDays(“$HOME/myapp/mylicense.lic”,”tree”,”4.0”,daysPtr);
days = PSProlibProxy.intPtr_value(daysPtr);
PSProlibProxy.delete_intPtr(daysPtr);
```

```
[Python]
retVal, days = PSProlibProxy.sgExpireDays(“$HOME/myapp/mylicense.lic”,”tree”,”4.0”)
```

```
[Visual Basic .NET]
Dim retVal As Integer
Dim days As Integer
retVal =  PSProlibProxy.sgExpireDays(“@mylicense.lic”,”tree”,”4.0”,days)
```

```
[PHP 5]
$daysPtr = new_intp();
sgExpireDays('$HOME/Sites/mylicense.lic','tree','4'0',$daysPtr);
$days =  intp_value($daysPtr);
delete_intp($daysPtr);
```

```
[MT5]
int retVal, days;
char target[256], component[16], version[10];
StringToCharArray("%HOME%\\MT5\\trademaster.lic",target,0,WHOLE_ARRAY,CP_UTF8);
StringToCharArray("trademaster",component,0,WHOLE_ARRAY,CP_UTF8);
StringToCharArray("5.0",version,0,WHOLE_ARRAY,CP_UTF8);
retVal = sgExpireDays(target,component,version,days);
```

```
[FORTRAN]
integer retVal,days
retVal = sgExpireDays('@mylicense.lic','tree','4.0',days)
```

Parameters

target [in]

The license file path for **SG_LICENSETYPE_EXTERNAL** licenses. The application path for **SG_LICENSETYPE_INTERNAL** licenses. For **SG_LICENSETYPE_EXTERNAL** licenses, you can specify “@mylicense.lic” for example. The “@” tells the software the license file resides next to the application executable. This makes it easy to locate a license file without having to construct a pathname. You may also use environment variables and relative pathnames here too for Windows, UNIX and Linux.

NOTE: Do not use the @ symbol for Java or Python, as the software will look for the license file next to the Java or Python binary, e.g, /usr/bin/java, not where your java code is located.

component [in]

The component name.

version [in]

The version number of the component

days [out]

The number of days until the license expires. A value of 10000 days indicates the license does not timeout

Description

Returns the number of days until the specified component expires. A value of 10,000 days indicates the license does not expire. This call is not valid when **SG_LICENSETYPE_FLOATING** is set.

Return codes

On success, **SG_SUCCESS** is returned.

On error, possible return codes are:

SG_NOT_AUTHORIZED_FOR_COMPONENT
SG_NOT_AUTHORIZED_FOR_VERSION
SG_COMPONENT_EXPIRED
SG_TARGET_FILE_NOT_FOUND
SG_CANT_READ_TARGET_FILE
SG_COMPONENT_NAME_TOO_LONG
SG_VERSION_LENGTH_TOO_LONG
SG_INVALID_OPTION_FOR_LICENSETYPE

Floating license implementation functions:

These are the API calls to implement floating licenses within your software.

sgConnect()

Make a connection to the license server.

[C/C++]
```
int retVal;
retVal = sgConnect();
```

[Objective C]
```
NSInteger retVal;
retVal = [PSProlibProxy sgConnect:self];
```

[Java]
```
int retVal;
retVal = PSProlibProxy.sgConnect();
```

[Python]
```
retVal = PSProlibProxy.sgConnect()
```

[Visual Basic .NET]
```
Dim retVal As Integer
retVal = PSProlibProxy.sgConnect()
```

[PHP 5]
```
$retVal = sgConnect();
```

[MT5]
```
int retVal;
retVal = sgConnect();
```

[FORTRAN]
```
integer retVal
retVal = sgConnect()
```

Parameters

none

Description

Establishes a connection with the *SafeGuard LM* License Server. This call is not required as all floating license functions call this internally. But it can be used to determine connectivity between the client and server without actually checking out a license. Once connected with **sgConnect()** and the connection state is no longer needed, a call to **sgDisconnect()** can be made but is not necessary.

Return codes

On success, **SG_SUCCESS** is returned

On error, possible return codes are:

SG_INVALID_OPTION_FOR_LICENSETYPE
SG_INVALIDHOST
SG_SERVERNORESPOND

SG_BADWRITE
SG_LOSTCONNECT
SG_BADREAD
SG_BADCOMMUNICATION
SG_BADCONNECT

sgDisconnect()

Disconnect from the license server.

```
[C/C++]
int retVal;
retVal = sgDisconnect();
```

```
[Objective C]
NSInteger retVal;
retVal = [PSProlibProxy sgDisconnect:self];
```

```
[Java]
int retVal;
retVal = PSProlibProxy.sgDisconnect();
```

```
[Python]
retVal = PSProlibProxy.sgDisconnect()
```

```
[Visual Basic .NET]
Dim retVal As Integer
retVal = PSProlibProxy.sgDisconnect()
```

```
[PHP 5]
$retVal = sgDisonnect();
```

```
[MT5]
int retVal;
retVal = sgDisconnect();
```

```
[FORTRAN]
integer retVal
retVal = sgDisconnect()
```

Parameters

none

Description

Disconnects from the *SafeGuard LM* License Server. This call is not required before your application terminates as the License Server will automatically detect the termination of your application and return any checked out licenses to the license pool. However, if you would like to control this for any reason, you can disconnect from the License Server and any checked out licenses will be checked in.

Return codes

On success, **SG_SUCCESS** is returned

On error, possible return codes are:

SG_INVALID_OPTION_FOR_LICENSETYPE
SG_LOSTCONNECT
SG_BADREAD

SG_BADCOMMUNICATION
SG_BADCONNECT

sgCheckout()

Check out a license from the license server.

```
[C/C++]
int retVal;
retVal = sgCheckout(“f1”,”4.0”);
```

```
[Objective C]
NSInteger retVal;
retVal = [PSProlibProxy sgCheckout:self forComponent:@”f1” forVersion:@”4.0”];
```

```
[Java]
int retVal;
retVal = PSProlibProxy.sgCheckout(“f1”,”4.0”);
```

```
[Python]
retVal = PSProlibProxy.sgCheckout(“f1”,”4.0”)
```

```
[Visual Basic .NET]
Dim retVal As Integer
retVal = PSProlibProxy.sgCheckout(“f1”,”4.0”)
```

```
[PHP 5]
$retVal = sgCheckout('f1','4.0');
```

```
[MT5]
int retVal;
char component[16],version[10];
StringToCharArray("trademaster",component,0,WHOLE_ARRAY,CP_UTF8);
StringToCharArray("5.0",version,0,WHOLE_ARRAY,CP_UTF8);
retVal = sgCheckout(component,version);
```

```
[FORTRAN]
integer retVal
retVal = sgCheckout('f1','4.0')
```

Parameters

component [in]

The licensed component name

version [in]

The minimum version desired

Description

Attempt to checkout the specified license and version.   The specified version must be exact. If the license server does not support the version requested, the return code will be **SG_NOT_LICENSED_FOR_VERSION**.

Note: licenses are checked out on a per call basis. So a single process “A” requesting license “f3” will checkout one “f3” license each time **sgCheckout()** is called. A call to **sgCheckin()** is not required when your application ends, as the license server will detect the broken connection and automatically release any checked out licenses.

Return codes

On success, **SG_SUCCESS** is returned

On error, possible return codes are:

SG_ALL_LICENSES_IN_USE
SG_NO_SUCH_COMPONENT
SG_NOT_LICENSED_FOR_VERSION
SG_BADWRITE
SG_LOSTCONNECT
SG_BADREAD
SG_BADCOMMUNICATION
SG_BADCONNECT
SG_INVALID_OPTION_FOR_LICENSETYPE

sgCheckin()

Return a license to the license server

```
[C/C++]
int retVal;
retVal = sgCheckin(“f1”,”4.0”);
```

```
[Objective C]
NSInteger retVal;
retVal = [PSProlibProxy sgCheckin:self forComponent:@”f1” forVersion:@”4.0”];
```

```
[Java]
int retVal;
retVal = PSProlibProxy.sgCheckin(“f1”,”4.0”);
```

```
[Python]
retVal = PSProlibProxy.sgCheckin(“f1”,”4.0”)
```

```
[Visual Basic .NET]
Dim retVal As Integer
retVal = PSProlibProxy.sgCheckin(“f1”,”4.0”)
```

```
[PHP 5]
$retVal = sgCheckin('f1','4.0');
```

```
[MT5]
int retVal;
char component[16],version[10];
StringToCharArray("trademaster",component,0,WHOLE_ARRAY,CP_UTF8);
StringToCharArray("5.0",version,0,WHOLE_ARRAY,CP_UTF8);
retVal = sgCheckin(component,version);
```

```
[FORTRAN]
integer retVal
retVal = sgCheckin('f1','4.0')
```

Parameters

component [in]

The component name to license pool

version [in]

The version to check in

Description

Check in or return the license to the license pool managed by the license server.

Return codes

On success, **SG_SUCCESS** is returned

On error, possible return codes are:

SG_LOSTCONNECT
SG_BADREAD
SG_BADCOMMUNICATION
SG_BADCONNECT
SG_NOT_CHECKED_OUT
SG_INVALID_OPTION_FOR_LICENSETYPE

sgTest()

Tests whether a license is available on the license server. No attempt to check out the license is made.

```
[C/C++]
int retVal;
retVal = sgTest(“f1”,”4.0”);
```

```
[Objective C]
NSInteger retVal;
retVal = [PSProlibProxy sgTest:self forComponent:@”f1” forVersion:@”4.0”];
```

```
[Java]
int retVal;
retVal = PSProlibProxy.sgTest(“f1”,”4.0”);
```

```
[Python]
retVal = PSProlibProxy.sgTest(“f1”,”4.0”)
```

```
[Visual Basic .NET]
Dim retVal As Integer
retVal = PSProlibProxy.sgTest(“f1”,”4.0”)
```

```
[PHP 5]
$retVal = sgTest('f1','4.0');
```

```
[MT5]
int retVal;
char component[16],version[12];
StringToCharArray("trademaster",component,0,WHOLE_ARRAY,CP_UTF8);
StringToCharArray("5.0",version,0,WHOLE_ARRAY,CP_UTF8);
retVal = sgTest(component,version);
```

```
[FORTRAN]
integer retVal
retVal = sgTest('f1','4.0')
```

Parameters

component [in]

The component name to check

version [in]

The version desired

Description

Test whether the specified component and version are served by the license server, not whether the component can be checked out. No attempt to check out the license is performed.

Return codes

On success, **SG_SUCCESS** is returned

On error, possible return codes are:

SG_SRV_NO_SUCH_COMPONENT
SG_NO_SUCH_COMPONENT
SG_NOT_LICENSED_FOR_VERSION
SG_BADWRITE
SG_LOSTCONNECT
SG_BADREAD
SG_BADCOMMUNICATION
SG_BADCONNECT
SG_INVALID_OPTION_FOR_LICENSETYPE

sgPing()

Ping the license server.

[C/C++]
int retVal;
retVal = sgPing();

[Objective C]
NSInteger retVal;
retVal = [PSProlibProxy sgPing:self];

[Java]
int retVal;
retVal = PSProlibProxy.sgPing();

[Python] **retVal = PSProlibProxy.sgPing()**
[Visual Basic .NET] **Dim retVal As Integer** **retVal = PSProlibProxy.sgPing()**
[PHP 5] **$retVal = sgPing();**

[MT5]
int retVal;
retVal = sgPing();

[FORTRAN] **integer retVal** **retVal = sgPing()**

Parameters

none

Description

Ping the license server. This is not the same as calling **sgTimer()**. **sgPing()** just makes sure the client can communicate with the license server. **sgPing()** will reconnect with the license server if the connection had been lost for any reason.

Return codes

On success, **SG_SUCCESS** is returned

On error, possible return codes are:

SG_LOSTCONNECT
SG_BADREAD
SG_BADCOMMUNICATION
SG_BADCONNECT
SG_INVALID_OPTION_FOR_LICENSETYPE

sgTimer()

Used to manually ensure the connection with the license server is maintained.

[C/C++]
int retVal;
retVal = sgTimer();

[Objective C]
NSInteger retVal;
retVal = [PSProlibProxy sgTimer:self];

[Java]
int retVal;
retVal = PSProlibProxy.sgTimer();

[Python] **retVal = PSProlibProxy.sgTimer()**
[Visual Basic .NET] **Dim retVal As Integer** **retVal = PSProlibProxy.sgTimer()**
[PHP 5] **$retVal = sgTimer();**

[MT5]
int retVal;
retVal = sgTimer();

[FORTRAN] **integer retVal** **retVal = sgTimer()**

Parameters

none

Description

sgTimer() can be utilized in situations where **alarm()** signals can't be tolerated (does not apply to Windows). The software will periodically attempt to make sure the connection to the license server is good. This periodic check is performed every two minutes by default. If your application cannot tolerate **alarm()** signals, then you can disable the automatic timer call using **sgSetAttrInt(SG_ATTR_DISABLE_ALARM,1)** before any connection to the license server is made, and call **sgTimer()** yourself. You should not call this more than once every minute or so if you decide to call it yourself. If you don't care about **alarm()** timers, just let *SafeGuard LM* do the check for you.

sgTimer() does nothing if there is no connection to the license server. If there is a connection to the license server and it is lost, **sgTimer()** will attempt a reconnect and if it is unable to reconnect, it will increment the attempts until **SG_ATTR_RETRY_COUNT** is reached. At which time it will terminate the application unless you have registered your own license manager callbacks through one of the **sgSetAttrFn*()** functions. For example, **sgSetAttrFnC()** for C/C++, **sgSetAttrFnP()** for Python, **sgSetAttrFnJ()** for Java, and so on. These callback functions let you handle the situation where the license server has either been shut down, killed or in general the connection between the client and server is no longer active but your application is still running. This function is a manual replacement for the default handlers that are already in place with *SafeGuard LM*. So, unless you really know you need to use this functionality, you probably don't.

sgTimer() invokes the callbacks set by the above functions to handle this scenario.

Return codes

On success, **SG_SUCCESS** is returned

On error, possible return codes are:

SG_INVALID_OPTION_FOR_LICENSETYPE

sgShutdown()

Shutdown the license server.

```
[C/C++]
int retVal;
retVal = sgShutdown();
```

```
[Objective C]
NSInteger retVal;
retVal = [PSProlibProxy sgShutdown:self];
```

```
[Java]
int retVal;
retVal = PSProlibProxy.sgShutdown();
```

```
[Python]
retVal = PSProlibProxy.sgShutdown()
```

```
[Visual Basic .NET]
Dim retVal As Integer
retVal = PSProlibProxy.sgShutdown()
```

```
[PHP 5]
$retVal = sgShutdown();
```

```
[MT5]
int retVal;
retVal = sgShutdown();
```

```
[FORTRAN]
integer retVal
retVal = sgShutdown()
```

Parameters

none

Description

Shuts down the License Server. This should not be called under normal circumstances. Use your own judgement.

Return codes

On success, **SG_SUCCESS** is returned

On error, possible return codes are:

SG_LOSTCONNECT
SG_BADREAD
SG_BADCOMMUNICATION
SG_BADCONNECT
SG_INVALID_OPTION_FOR_LICENSETYPE

sgRestart()

Restart the license server.

[C/C++]
int retVal;
retVal = sgRestart();

[Objective C]
NSInteger retVal;
retVal = [PSProlibProxy sgRestart:self];

[Java]
int retVal;
retVal = PSProlibProxy.sgRestart();

[Python] **retVal = PSProlibProxy.sgRestart()**
[Visual Basic .NET] **Dim retVal As Integer** **retVal = PSProlibProxy.sgRestart()**
[PHP 5] **$retVal = sgRestart();**

[MT5]
int retVal;
retVal = sgRestart();

[FORTRAN] **integer retVal** **retVal = sgRestart()**

Parameters

none

Description

Restarts the License Server. This should not be called under normal circumstances unless you are updating the license file and wish to restart the server. Use your own judgement.

Return codes

On success, **SG_SUCCESS** is returned

On error, possible return codes are:

SG_LOSTCONNECT
SG_BADREAD
SG_BADCOMMUNICATION
SG_BADCONNECT
SG_INVALID_OPTION_FOR_LICENSETYPE

sgShowClient()

Retrieves information about what components the current application has checked out.

```
[C/C++]
int retVal, count;
char serverhost[256],clienthost[256],user[32],component[16],version[12];
retVal = sgShowClient(serverhost,clienthost,user,component,version,&count);
```

```
[Objective C]
NSInteger retVal, count;
NSString *serverhost,*clienthost,*user,*component,*version;
retVal = [PSProlibProxy sgShowClient:self getServer:&serverhost getClient:&clienthost getUser:&user
                                          getComponent:&component getVersion:&version getCount:&count];
```

```
[Java]
int retVal, count;
SWIGTYPE_p_int countPtr;
byte serverhost[] = new byte[256];
byte clienthost[] = new byte[256];
byte user[] = new byte[32];
byte component[] = new byte[16];
byte version[] = new byte[12];
// create the int pointer
countPtr = PSProlibProxy.new_intPtr();
retVal = PSProlibProxy.sgShowClient(serverhost,clienthost,user,component,version,countPtr);
// dereference the int pointer
count = PSProlibProxy.intPtr_value(countPtr);
// delete the int pointer
PSProlibProxy.delete_intPtr(countPtr);
```

```
[Python]
retVal, serverhost, clienthost, user, component, version, count = PSProlibProxy.sgShowClient()
```

```
[Visual Basic .NET]
Dim retVal As Integer
Dim count As Integer
Dim serverhost As New StringBuilder(256)
Dim clienthost As New StringBuilder(256)
Dim user As New StringBuilder(32)
Dim component As New StringBuilder(16)
Dim version As New StringBuilder(12)
retVal =  PSProlibProxy.sgShowClient(serverhost,clienthost,user,component,version,count)
```

```
[PHP 5]
$countPtr = new_intp();
$serverhost = str_repeat(' ',256);
$clienthost = str_repeat(' ',256);
$user = str_repeat(' ',32);
$component = str_repeat(' ',16);
$version = str_repeat(' ',12);
$retVal = sgShowClient($serverhost,$clienthost,$user,$component,$version,$countPtr);
$count =  intp_value($countPtr);
```

[MT5]
```
int retVal,count;
char serverhost[256], clienthost[256], user[32], component[16], version[12];
retVal = sgShowClient(serverhost, clienthost, user, component, version, count);
```

[FORTRAN]
```
integer retVal, count
character*256 serverhost, clienthost
character*32 user
character*16 component
character*12 version
retVal = sgShowClient(serverhost,clienthost,user,component,version,count)
```

Parameters

serverhost [out]

The server hostname

clienthost [out]

The client hostname

user [out]

The username

component [out]

The component name.

version [out]

The version number of the component

count [out]

The number of licenses checked out for each component

Description

Retrieves information about what components the client has checked out. For example, you would use this function to retrieve the users licenses and display them however appropriate. This function can be called until **SG_NO_MORE_CLIENT_COMPONENTS** is returned. The next time it is called, it will start at the beginning.

Return codes

On success, **SG_SUCCESS** is returned.

On error, possible return codes are:

SG_INVALID_OPTION_FOR_LICENSETYPE

SG_NO_MORE_CLIENT_COMPONENTS
SG_LOSTCONNECT
SG_BADREAD
SG_BADCOMMUNICATION
SG_BADCONNECT

sgShowServer()

Retrieves information about what components the license server has available.

[C/C++]
```
int retVal,count,current;
char component[16],version[12],hostid[256],idtype[16],expires[12],certificate[128];
retVal = sgShowServer(component,version,hostid,idtype,&count,&current,expires,certificate);
```

[Objective C]
```
NSInteger retVal,count,current;
NSString *component,*version,*hostid,*idstring,*idtype,*expires,*certificate;
retVal = [PSProlibProxy sgShowServer:self getComponent:&component getVersion:&version getId:&hostid
    getIdType:&idtype getCount:&count getCurrent:&current getExpires:&expires getCertificate:&certificate];
```

[Java]
```
int retVal,count,current;
SWIGTYPE_p_int countPtr;
SWIGTYPE_p_int currentPtr;
byte component[] = new byte[16];
byte version[] = new byte[12];
byte hostid[] = new byte[256];
byte idtype[] = new byte[16];
byte expires[] = new byte[12];
byte certificate[] = new byte[128];
// create the int pointers
countPtr = PSProlibProxy.new_intPtr();
currentPtr = PSProlibProxy.new_intPtr();
retVal = PSProlibProxy.sgShowServer(component,version,hostid,idtype,countPtr,currentPtr,expires,certificate);
// dereference the int pointers
count = PSProlibProxy.intPtr_value(countPtr);
current = PSProlibProxy.intPtr_value(currentPtr);
// delete the int pointers
PSProlibProxy.delete_intPtr(countPtr);
PSProlibProxy.delete_intPtr(currentPtr);
```

[Python]
```
retVal, component,version,hostid,idtype,count,current,expires,certificate = PSProlibProxy.sgShowServer()
```

[Visual Basic .NET]
```
Dim retVal As Integer
Dim count As Integer
Dim current As Integer
Dim component As New StringBuilder(16)
Dim version As New StringBuilder(12)
Dim hostid As New StringBuilder(256)
Dim idtype As New StringBuilder(16)
Dim expires As New StringBuilder(12)
Dim certificate As New StringBuilder(128)
retVal =  PSProlibProxy.sgShowServer(component,version,hostid,idtype,count,current,expires,certificate)
```

[PHP 5]
```
$countPtr = new_intp()
$currentPtr = new_intp();
```

```
$component = str_repeat(' ',16);
$version = str_repeat(' ',12);
$hostid = str_repeat(' ',256);
$idtype = str_repeat(' ',16);
$expires = str_repeat(' ',12);
$certificate = str_repeat(' ',128);
$retVal = sgShowServer($component,$version,$hostid,$idtype,$countPtr,$currentPtr,$expires,$certificate);
$count =  intp_value($countPtr);
$current =  intp_value($currentPtr);
delete_intp($countPtr);
delete_intp($currentPtr);
```

[MT5]

```
int retVal, count, current;
char component[16], version[12], hostid[256], idtype[16], expires[12], certificate[128];
retVal = sgShowServer(component, version, hostid, idtype, count, current, expires, certificate);
```

[FORTRAN]

```
integer retVal,count,current
character*256 hostid
character*128 certificate
character*16 component, idtype
character*12 version,expires
retVal = sgShowServer(component,version,hostid,idtype,count,current,expires,certificate)
```

Parameters

component [out]

The component name

version [out]

The version of the component

hostid [out]

The *SafeGuard LM* host ID

idtype [out]

The idtype string

count [out]

The total number of licenses

current [out]

The currently available licenses

expires [out]

The expiration date. A date of “**never”** means the component does not expire certificate [out]

The component line certificate which can be in either **RC4** or **SHA256** format.

Description

Retrieves the next licensed component from the License Server. This functions purpose is to retrieve component licenses for display purposes. This function can be called until **SG_NO_MORE_SERVER_COMPONENTS** is returned. However, if you call this after the error, it will start at the top of the list. A great way for real time updating of license status.

Return codes

On success, **SG_SUCCESS** is returned.

On error, possible return codes are:

SG_INVALID_OPTION_FOR_LICENSETYPE
SG_NO_MORE_SERVER_COMPONENTS
SG_LOSTCONNECT
SG_BADREAD
SG_BADCOMMUNICATION
SG_BADCONNECT

sgShowServerDetail()

Retrieves information about everything that is checked out from the license server.

[C/C++]
```
int retVal,thread_id,sockfd,pid,count;
char ip[16],process[256],user[128],host[256],component[32],version[12];
retVal = sgShowServerDetail(&thread_id,&sockfd,ip,&pid,process,user,host,component,version,&count);
```

[Objective C]
```
NSInteger retVal,thread_id,sockfd,pid,count;
NSString *ip,*process,*user,*host,*component,*version;
retVal = [PSProlibProxy sgShowServerDetail:self getThreadID:&thread_id getSocket:&sockfd getIP:&ip
        getPID:&pid getProcess:&process getUser:&user getHost:&host getComponent:&component
getVersion:&version getCount:&count];
```

[Java]
```
int retVal,thread_id,sockfd,pid,count;
SWIGTYPE_p_int thread_idPtr;
SWIGTYPE_p_int sockfdPtr;
SWIGTYPE_p_int pidPtr;
SWIGTYPE_p_int countPtr;
byte ip[] = new byte[16];
byte process[] = new byte[256];
byte user[] = new byte[128];
byte host[] = new byte[256];
byte component[] = new byte[32];
byte version[] = new byte[12];
// create the int pointers
thread_idPtr = PSProlibProxy.new_intPtr();
sockfdPtr = PSProlibProxy.new_intPtr();
pidPtr = PSProlibProxy.new_intPtr();
countPtr = PSProlibProxy.new_intPtr();
retVal = PSProlibProxy.sgShowServerDetail(thread_idPtr,sockfdPtr,ip,pidPtr,process,user,host,
                            component,version,countPtr);
// dereference the int pointers
thread_id = PSProlibProxy.intPtr_value(thread_idPtr);
sockfd = PSProlibProxy.intPtr_value(sockfdPtr);
pid = PSProlibProxy.intPtr_value(pidPtr);
count = PSProlibProxy.intPtr_value(countPtr);
// delete the int pointers
PSProlibProxy.delete_intPtr(thread_idPtr);
PSProlibProxy.delete_intPtr(sockfdPtr);
PSProlibProxy.delete_intPtr(pidPtr);
PSProlibProxy.delete_intPtr(countPtr);
```

[Python]
```
retVal,thread_id,sockfd,ip,pid,process,user,host,component,version,count  = PSProlibProxy.sgShowServerDetail()
```

[Visual Basic .NET]
```
Dim retVal As Integer
Dim thread_id As Integer
Dim sockfd As Integer
Dim pid As Integer
Dim count As Integer
Dim ip As New StringBuilder(16)
Dim process As New StringBuilder(256)
Dim userAs New StringBuilder(128)
Dim host As New StringBuilder(256)
Dim component As New StringBuilder(32)
Dim version As New StringBuilder(12)
retVal =  PSProlibProxy.sgShowServerDetail(thread_id,sockfd,ip,pid,process,user,host,component,version,count)
```

[PHP 5]

```
$thread_idPtr = new_intp();
$sockfdPtr = new_intp();
$pidPtr = new_intp();
$countPtr = new_intp();
$ip = str_repeat(' ',16);
$process = str_repeat(' ',256);
$user = str_repeat(' ',128);
$host = str_repeat(' ',256);
$component = str_repeat(' ',32);
$version = str_repeat(' ',12);
$retVal = sgShowServerDetail($thread_idPtr,$sockfdPtr,$ip,$pidPtr,$process,$user,$host,$component,$version,
$countPtr);
$thread_id =  intp_value($thread_idPtr);
$sockfd =  intp_value($sockfdPtr);
$pid = intp_value($pidPtr);
count = intp_value($countPtr);
delete_intp($thread_idPtr);
delete_intp($sockfdPtr);
delete_intp($pidPtr);
delete_intp($countPtr);
```

[MT5]

```
int retVal, thread_id,sockfd,pid,count;
char ip[16],process[256],user[128],host[256],component[32],version[12];
retVal = sgShowServerDetail(thread_id,sockfd,ip,pid,process,user,host,component,version,count);
```

[FORTRAN]

```
integer retVal,thread_id,sockfd,pid,count
character*256 process,host
character*128 user
character*16 component,ip
character*12 version
retVal = sgShowServerDetail(thread_id,sockfd,ip,pid,process,user,host,component,version,count)
```

Parameters

thread_id [out]

The internal thread ID for the checked out license

sockfd [out]

The socket file descriptor for the client connection ip [out]

The client IP address

pid [out]

The process ID (PID) of the client application

process [out]

The pathname of the client process

user [out]

The username of the person who checked out the license

host [out]

The hostname where the license was checked out

component [out]

The component name

version [out]

The version number of the component checked out

count [out]

The number of licenses checked out for this component by this user

Description

Retrieves the next checked out component from the License Server. This functions purpose is to retrieve detailed information regarding all checked out licenses. This function can be called until **SG_NO_MORE_SERVER_DETAIL** is returned. However, if you call this after the error, it will start at the top of the list. A great way for real time updating of license status.

Return codes

On success, **SG_SUCCESS** is returned.

On error, possible return codes are:

SG_INVALID_OPTION_FOR_LICENSETYPE
SG_NO_MORE_SERVER_DETAIL
SG_LOSTCONNECT
SG_BADREAD
SG_BADCOMMUNICATION
SG_BADCONNECT

sgSetAttrFnC()

Sets floating license callbacks for when a connection to the license server is lost. This is for C/C++ language interfaces.

```
[C/C++]
int myRecon(char *server_name,int retry_attempt, int total_retry_count);
int myReconCheckoutFail(char *server_name,char *component, char *version);
int myReconComplete(char *server_name,int retry_attempt);
int myReconExit(char *server_name,int retry_attempt);

int retVal;
retVal = sgSetAttrFnC(SG_ATTR_VENDOR_RECONNECT, myRecon);
retVal = sgSetAttrFnC(SG_ATTR_VENDOR_RECONNECT_CHECKOUT_FAIL, myReconCheckoutFail);
retVal = sgSetAttrFnC(SG_ATTR_VENDOR_RECONNECT_COMPLETE, myReconComplete);
retVal = sgSetAttrFnC(SG_ATTR_VENDOR_RECONNECT_EXIT, myReconExit);
```

Parameters

attrtype [in]

SG_ATTR_VENDOR_RECONNECT_EXIT
SG_ATTR_VENDOR_RECONNECT
SG_ATTR_VENDOR_RECONNECT_CHECKOUT_FAIL
SG_ATTR_VENDOR_RECONNECT_COMPLETE

attrval [in]

The function pointer

Description

These are for floating licenses only. These define the function callbacks for when a connection to the license server is lost. **SG_ATTR_VENDOR_RECONNECT** is called once every **SG_ATTR_RETRY_INTERVAL** minutes up to **SG_ATTR_RETRY_COUNT** times. If during those times a connection is re-established with the license server, then **SG_ATTR_VENDOR_RECONNECT_COMPLETE** is called. If a connection cannot be re-established after **SG_ATTR_RETRY_COUNT** times, then **SG_ATTR_VENDOR_RECONNECT_EXIT** is called and it is up to you to decide what to do with the application. If a connection is re-established, but an existing license cannot be checked out for whatever reason, then **SG_ATTR_VENDOR_RECONNECT_CHECKOUT_FAIL** is called for each component that cannot be re-checked out. If you do not define these functions in your application, the default behavior is to print information regarding the reconnect attempts to *stdout*, and if a connection cannot be re-established, the app is terminated.

Return codes

On success, **SG_SUCCESS** is returned.

On error, possible return codes are:

SG_ATTRTYPE_NOT_SUPPORTED

sgSetAttrFnF()

Sets floating license callbacks when a connection to the license server is lost. This is for the FORTRAN language interface

```
[FORTRAN]
      external myRecon,myReconCheckoutFail,myReconComplete
      external myReconExit
      integer myRecon,myReconCheckoutFail,myReconComplete
      integer myReconExit

integer retVal
retVal = sgSetAttrFnF(SG_ATTR_VENDOR_RECONNECT, myRecon)
retVal = sgSetAttrFnF(SG_ATTR_VENDOR_RECONNECT_CHECKOUT_FAIL, myReconCheckoutFail)
retVal = sgSetAttrFnF(SG_ATTR_VENDOR_RECONNECT_COMPLETE, myReconComplete)
retVal = sgSetAttrFnF(SG_ATTR_VENDOR_RECONNECT_EXIT, myReconExit)
```

Parameters

attrtype [in]

SG_ATTR_VENDOR_RECONNECT_EXIT
SG_ATTR_VENDOR_RECONNECT
SG_ATTR_VENDOR_RECONNECT_CHECKOUT_FAIL
SG_ATTR_VENDOR_RECONNECT_COMPLETE

attrval [in]

The function pointer

Description

See Floating.f in the fortran folder under the examples folder for example callback functions

These are for floating licenses only. These define the function callbacks for when a connection to the license server is lost. **SG_ATTR_VENDOR_RECONNECT** is called once every **SG_ATTR_RETRY_INTERVAL** minutes up to **SG_ATTR_RETRY_COUNT** times. If during those times a connection is re-established with the license server, then **SG_ATTR_VENDOR_RECONNECT_COMPLETE** is called. If a connection cannot be re-established after **SG_ATTR_RETRY_COUNT** times, then **SG_ATTR_VENDOR_RECONNECT_EXIT** is called and it is up to you to decide what to do with the application. If a connection is re-established, but an existing license cannot be checked out for whatever reason, then **SG_ATTR_VENDOR_RECONNECT_CHECKOUT_FAIL** is called for each component that cannot be re-checked out. If you do not define these functions in your application, the default behavior is to print information regarding the reconnect attempts to *stdout*, and if a connection cannot be re-established, the app is terminated.

Return codes

On success, **SG_SUCCESS** is returned.

On error, possible return codes are:

SG_ATTRTYPE_NOT_SUPPORTED

sgSetAttrFnP()

Sets floating license callbacks for when a connection to the license server is lost. This is for the Python language interface.

```
[Python]
import PSProlibProxy

def recon(server,retries,count):
	print "in reconnect:",server,retries,count

def recon_complete(server,retries):
	print "in reconnect complete:",server,retries

def recon_exit(server,retries):
	print "in reconnect exit:",server,retries

def recon_checkout_fail(server,component,version):
	print "in reconnect checkout fail:",server,component,version

PSProlibProxy.sgSetAttrFnP(psprolib.SG_ATTR_VENDOR_RECONNECT,recon)
PSProlibProxy.sgSetAttrFnP(psprolib.SG_ATTR_VENDOR_RECONNECT_CHECKOUT_FAIL,recon_checkout
_fail)
PSProlibProxy.sgSetAttrFnP(psprolib.SG_ATTR_VENDOR_RECONNECT_COMPLETE,recon_complete)
PSProlibProxy.sgSetAttrFnP(psprolib.SG_ATTR_VENDOR_RECONNECT_EXIT,recon_exit)
```

Parameters

attrtype [in]

SG_ATTR_VENDOR_RECONNECT_EXIT
SG_ATTR_VENDOR_RECONNECT
SG_ATTR_VENDOR_RECONNECT_CHECKOUT_FAIL
SG_ATTR_VENDOR_RECONNECT_COMPLETE

attrval [in]

The function name

Description

These are for floating licenses only. These define the function callbacks for when a connection to the license server is lost. **SG_ATTR_VENDOR_RECONNECT** is called once every **SG_ATTR_RETRY_INTERVAL** minutes up to **SG_ATTR_RETRY_COUNT** times. If during those times a connection is re-established with the license server, then **SG_ATTR_VENDOR_RECONNECT_COMPLETE** is called. If a connection cannot be re-established after **SG_ATTR_RETRY_COUNT** times, then **SG_ATTR_VENDOR_RECONNECT_EXIT** is called and it is up to you to decide what to do with the application. If a connection is re-established, but an existing license cannot be checked out for whatever reason, then **SG_ATTR_VENDOR_RECONNECT_CHECKOUT_FAIL** is called for each component that cannot be re-checked out. If you do not define these functions in your application, the default behavior is to print information regarding the reconnect attempts to *stdout*, and if a connection cannot be re-established, the app is terminated.

Return codes

On success, **SG_SUCCESS** is returned.

On error, possible return codes are:

SG_ATTRTYPE_NOT_SUPPORTED

sgSetAttrFnJ()

Sets floating license callbacks for when a connection to the license server is lost. This is for the Java language interface.

```
[Java]
class sgLMCallbacks {
  private static void vendor_reconnect( String server, int retries, int count) {
     System.out.println("Java: Lost connection with server " + server + ". Attempting to reconnect " + retries + " of
" + count + " ...");
  }
  private static void vendor_reconnect_complete( String server, int retries) {
     System.out.println("Java: Reconnected with server " + server + " after " + retries + " attempts ...");
  }
  private static void vendor_reconnect_checkout_fail( String server, String component, String version) {
     System.out.println("Java: Unable to check out license " + component + ", version " + version + ", from server "
+ server + "...");
     /* disable that functionality or pause and make another attempt here */
  }
  private static void vendor_reconnect_exit( String server, int retries) {
     System.out.println("Java: Maximum reconnect attempts of " + retries + " to server " + server + " ...");
     System.exit(1);
  }
}

PSProlibProxy.sgSetAttrFnJ(SG_ATTRTYPE.SG_ATTR_VENDOR_RECONNECT,"vendor_reconnect");
PSProlibProxy.sgSetAttrFnJ(SG_ATTRTYPE.SG_ATTR_VENDOR_RECONNECT_COMPLETE,"vendor_recon
nect_complete");
PSProlibProxy.sgSetAttrFnJ(SG_ATTRTYPE.SG_ATTR_VENDOR_RECONNECT_EXIT,"vendor_reconnect_ex
it");
PSProlibProxy.sgSetAttrFnJ(SG_ATTRTYPE.SG_ATTR_VENDOR_RECONNECT_CHECKOUT_FAIL,"vendor
_reconnect_checkout_fail");
```

Parameters

attrtype [in]

SG_ATTR_VENDOR_RECONNECT_EXIT
SG_ATTR_VENDOR_RECONNECT
SG_ATTR_VENDOR_RECONNECT_CHECKOUT_FAIL
SG_ATTR_VENDOR_RECONNECT_COMPLETE

attrval [in]

The double quoted method name

Description

These are for floating licenses only. These define the function callbacks for when a connection to the license server is lost. **SG_ATTR_VENDOR_RECONNECT** is called once every **SG_ATTR_RETRY_INTERVAL** minutes up to **SG_ATTR_RETRY_COUNT** times. If during those times a connection is re-established with the license server, then **SG_ATTR_VENDOR_RECONNECT_COMPLETE** is called. If a connection cannot be re-established after

SG_ATTR_RETRY_COUNT times, then **SG_ATTR_VENDOR_RECONNECT_EXIT** is called and it is up to you to decide what to do with the application. If a connection is re-established, but an existing license cannot be checked out for whatever reason, then **SG_ATTR_VENDOR_RECONNECT_CHECKOUT_FAIL** is called for each component that cannot be re-checked out. If you do not define these functions in your application, the default behavior is to print information regarding the reconnect attempts to *stdout*, and if a connection cannot be re-established, the app is terminated.

Note: the **sgLMCallbacks** class must be named just that for the four callback methods.

Return codes

On success, **SG_SUCCESS** is returned.

On error, possible return codes are:

SG_ATTRTYPE_NOT_SUPPORTED

Product Activation Server implementation functions:

If you wish to utilize the Product Activation Server within your application you would use the following API calls to do so.

sgPaActivate()

Obtain either a “demo” or “paid” license from the Product Activation Server over the Internet.

```
[C/C++]
int retVal;
char serverline[256],componentline[1024];
retVal = sgPaActivate(“www.yourcompany.com”,29775,SG_ACTIVATE_DEMO_NODE,NULL,”f1”,”4.0”,NULL,
                        serverline,componentline);
```

```
[Objective C]
NSInteger retVal;
NSString *serverline,*componentline;
retVal = [PSProlibProxy sgPaActivate:self forWeb:@"98.120.32.50" forPort:29775
         forActivate:SG_ACTIVATE_DEMO_NODE forCust:NULL forComponent:@"f1" forVersion:@"5.0"
         forEmail:NULL getServerLine:&serverline getComponentLine:&componentline]
```

```
[Java]
int retVal;
byte serverline[] = new byte[256];
byte componentline[] = new byte[1024];
retVal = PSProlibProxy.sgPaActivate(“www.yourcompany.com”,29775,
    SG_ACTIVATETYPE.SG_ACTIVATE_DEMO_NODE,NULL,”f1”,”4.0”,NULL,serverline,componentline);
```

```
[Python]
retVal, serverline,componentline = PSProlibProxy.sgPaActivate(“192.168.1.120”,29775,
                                   PSProlibProxy.SG_ACTIVATE_DEMO_NODE,NULL,”f1”,”4.0”,NULL)
```

```
[Visual Basic .NET]
Dim retVal As Integer
Dim serverline As New StringBuilder(256)
Dim componentline As New StringBuilder(1024)
retVal = PSProlibProxy.sgPaActivate(“www.yourcompany.com”,29775,
                                  psprolib.SG_ACTIVATETYPE.SG_ACTIVATE_DEMO_NODE,NULL,”f1”,”4.0”,
                                  NULL,serverline,componentline)
```

```
[PHP 5]
$serverline = str_repeat(' ',256);
$componentline = str_repeat(' ',1024);
$retVal = sgPaActivate('www.yourcompany.com',29775,SG_ACTIVATE_DEMO_NODE,
                        '','f1','4.0','userid@mail.com',$serverline,$componentline);
```

```
[MT5]
int retVal;
char serverline[256],componentline[1024],web[256],component[16],customer[256],version[12],email[256];
StringToCharArray("www.yourcompany.com",web,0,WHOLE_ARRAY,CP_UTF8);
StringToCharArray("trademaster",component,0,WHOLE_ARRAY,CP_UTF8);
StringToCharArray("5.0",version,0,WHOLE_ARRAY,CP_UTF8);
StringToCharArray("",customer,0,WHOLE_ARRAY,CP_UTF8);
StringToCharArray("user@example.com",email,0,WHOLE_ARRAY,CP_UTF8);
retVal = sgPaActivate(web,29775,SG_ACTIVATE_DEMO_NODE,customer,component,version,email,
                        serverline,componentline);
```

```
[FORTRAN]
integer retVal
character*1024 componentline
```

```
character*256 serverline
retVal = sgPaActivate('www.yourcompany.com',29775,SG_ACTIVATE_DEMO_NODE,NULL,'f1','4.0','',
                                 serverline,componentline)
```

Parameters

web [in]

The IP address or name of the activation server. i.e. **www.yourcompany.com**

port [in]

The TCP/IP port number of the activation server or **sgproxyserver** is listening on

activate [in]

The type of activation. Possible activation types are:

SG_ACTIVATE_DEMO_NODE	**SG_ACTIVATE_DEMO_FLOAT**
SG_ACTIVATE_PAID_NODE	**SG_ACTIVATE_PAID_FLOAT**

customer [in]

The unique identifier for “paid” customers. If the authorization record is in the SQL table with that unique customer number the license is fulfilled. If you specify **NULL** as the unique customer number, then the Ethernet (MAC) address is assumed to be the customer number and will be passed to the Product Activation Server automatically

component [in]

The desired component name

version [in]

The desired version number

email [in]

The email address of the customer. This can actually be anything you wish to send back to the Product Activation Server

outsrv [out]

The SERVER line for floating licenses. This is set only when **SG_ACTIVATE_DEMO_FLOAT** or **SG_ACTIVATE_PAID_FLOAT** is specified. Otherwise, it is blank

outlic [out]

The returned component line license from the Product Activation Server

Description

Used to obtain either a “demo” or “paid” license from the Product Activation Server over the Internet. The Product

Activation Server must be running and accessible and configured for the requested license to be successfully retrieved. Refer to the Product Activation Server documentation on how to configure and run. This function call will retrieve the necessary component line for your application to run. It must be properly installed once retrieved.

Return codes

On success, **SG_SUCCESS** is returned.

On error, possible return codes are:

SG_INVALIDHOST
SG_SERVERNORESPOND
SG_BADWRITE
SG_LOSTCONNECT
SG_BADREAD
SG_BADCOMMUNICATION
SG_BADCONNECT
SG_TIMEDIFF_TOO_LARGE
SG_COMPONENT_NOT_AVAILABLE
SG_COMPONENT_ALREADY_ISSUED
SG_CANT_QUERY_SQLDB
SG_CANT_CONNECT_TO_SQLDB
SG_WRONG_SERVER
SG_REJECTED
SG_COMPONENT_NAME_TOO_LONG
SG_VERSION_LENGTH_TOO_LONG

sgPaShutdown()

Used to remotely shutdown the Product Activation Server.

```
[C/C++]
int retVal;
retVal = sgPaShutdown(“www.yourcompany.com”,29775,”xyzzy”);
```

```
[Objective C]
NSInteger retVal;
retVal = [PSProlibProxy sgPaShutdown:self forWeb:@“www.yourcompany.com” forPort:29775
                                            forPassword:@”xyzzy”];
```

```
[Java]
int retVal;
retVal = PSProlibProxy.sgPaShutdown(“www.yourcompany.com”,29775,”xyzzy”);
```

```
[Python]
retVal  = PSProlibProxy.sgPaShutdown(“www.yourcompany.com”,29775,”xyzzy”)
```

```
[Visual Basic .NET]
Dim retVal As Integer
retVal = PSProlibProxy.sgPaShutdown(“www.yourcompany.com”,29775,”xyzzy”)
```

```
[PHP 5]
$retVal = sgPaShutdown('www.yourcompany.com',29775,'xyzzy');
```

```
[MT5]
int retVal;
char web[256],password[64];
StringToCharArray("www.yourcompany.com",web,0,WHOLE_ARRAY,CP_UTF8);
StringToCharArray("xyzzy",password,0,WHOLE_ARRAY,CP_UTF8);
retVal = sgPaShutdown(web,29775,password);
```

```
[FORTRAN]
integer retVal
retVal = sgPaShutdown('www.yourcompany.com',29775,'xyzzy')
```

Parameters

web [in]

The IP address or name of the activation server. i.e. **www.yourcompany.com**

port [in]

The TCP/IP port number the activation server or **sgproxyserver** is listening on

password [in]

The password set in the control.txt configuration file

Description

Used to remotely shutdown the Product Activation Server.

Return codes

On success, **SG_SUCCESS** is returned.

On error, possible return codes are:

SG_INVALIDHOST
SG_SERVERNORESPOND
SG_BADWRITE
SG_LOSTCONNECT
SG_BADREAD
SG_BADCOMMUNICATION
SG_BADCONNECT
SG_BAD_PASSWORD

sgPaRestart()

Used to remotely restart the Product Activation Server.

```
[C/C++]
int retVal;
retVal = sgPaRestart(“www.yourcompany.com”,29775,”xyzzy”);
```

```
[Objective C]
NSInteger retVal;
retVal = [PSProlibProxy sgPaRestart:self forWeb:@“www.yourcompany.com” forPort:29775
                                                        forPassword:@”xyzzy”];
```

```
[Java]
int retVal;
retVal = PSProlibProxy.sgPaRestart(“www.yourcompany.com”,29775,”xyzzy”);
```

```
[Python]
retVal  = PSProlibProxy.sgPaRestart(“www.yourcompany.com”,29775,”xyzzy”)
```

```
[Visual Basic .NET]
Dim retVal As Integer
retVal = PSProlibProxy.sgPaRestart(“www.yourcompany.com”,29775,”xyzzy”)
```

```
[PHP 5]
$retVal = sgPaRestart('www.yourcompany.com',29775,'xyzzy');
```

```
[MT5]
int retVal;
char web[256],password[64];
StringToCharArray("www.yourcompany.com",web,0,WHOLE_ARRAY,CP_UTF8);
StringToCharArray("xyzzy",password,0,WHOLE_ARRAY,CP_UTF8);
retVal = sgPaRestart(web,29775,password);
```

```
[FORTRAN]
integer retVal
retVal = sgPaRestart('www.yourcompany.com',29775,'xyzzy')
```

Parameters

web [in]

The IP address or name of the activation server. i.e. **www.yourcompany.com**

port [in]

The TCP/IP port number the activation server or **sgproxyserver** is listening on

password [in]

The password set in the control.txt configuration file

Description

Used to remotely restart the Product Activation Server.

Return codes

On success, **SG_SUCCESS** is returned.

On error, possible return codes are:

SG_INVALIDHOST
SG_SERVERNORESPOND
SG_BADWRITE
SG_LOSTCONNECT
SG_BADREAD
SG_BADCOMMUNICATION
SG_BADCONNECT
SG_BAD_PASSWORD

sgPaShowlog()

Used to get a copy of the Product Activation Server log file whether on the local machine or a remote system.

```
[C/C++]
int retVal;
char logfile[256];
retVal = sgPaShowlog("www.yourcompany.com",29775,"xyzzy",logfile);
```

```
[Objective C]
NSInteger retVal;
NSString *logfile;
retVal = [PSProlibProxy sgPaShowlog:self forWeb:@"www.yourcompany.com" forPort:29775
                                        forPassword:@"xyzzy" getLogfile:&logfile];
```

```
[Java]
int retVal;
byte logfile[] = new byte[256];
retVal = PSProlibProxy.sgPaShowlog("www.yourcompany.com",29775,"xyzzy",logfile);
```

```
[Python]
retVal, logfile  = PSProlibProxy.sgPaShowlog("www.yourcompany.com",29775,"xyzzy")
```

```
[Visual Basic .NET]
Dim retVal As Integer
Dim logfile As New StringBuilder(256)
retVal = PSProlibProxy.sgPaShowlog("www.yourcompany.com",29775,"xyzzy",logfile)
```

```
[PHP 5]
$logfile = str_repeat(' ',256);
$retVal = sgPaShowlog('www.yourcompany.com',29775,'xyzzy',$logfile);
```

```
[MT5]
int retVal;
char web[256],password[64],logfile[256];
StringToCharArray("www.yourcompany.com",web,0,WHOLE_ARRAY,CP_UTF8);
StringToCharArray("xyzzy",password,0,WHOLE_ARRAY,CP_UTF8);
retVal = sgPaShowlog(web,29775,password,logfile);
```

```
[FORTRAN]
integer retVal
character*256 logfile
retVal = sgPaShowlog('www.yourcompany.com',29775,'xyzzy',logfile)
```

Parameters

web [in]

The IP address or name of the activation server. i.e. **www.yourcompany.com**

port [in]

The TCP/IP port number the activation server or **sgproxyserver** is listening on

password [in]

The password set in the control.txt configuration file

logfile [out]

The pathname to a copy of the Product Activation Server log file.

Description

Used to get a copy of the currently running Product Activation Server log file. The server must have been started with the debug log file enabled. This filename will be saved most likely in /tmp or %TEMP% depending on your operating system. It is your duty to delete it when finished with it. It is an ASCII text file.

Return codes

On success, **SG_SUCCESS** is returned.

On error, possible return codes are:

SG_INVALIDHOST
SG_SERVERNORESPOND
SG_BADWRITE
SG_LOSTCONNECT
SG_BADREAD
SG_BADCOMMUNICATION
SG_BADCONNECT
SG_BAD_PASSWORD
SG_NOLOG

Product Activation Server (Alternate method):

The following API's are used if you wish to provide an alternative view of what a component line looks like to the end-user during the authorization process. The end result is exactly the same. They get a normal *SafeGuard LM* license. The only difference is how it is presented to the end-user.

These functions are a layer on top of **sgPaActivate()**. All they are really doing is hiding the different pieces of a SafeGuard LM component line in the form of **XXX-YYYY-ZZZZ** format. It can sometimes make it easier for Windows users to deal with this type of presentation.

sgPaGetInstallationCode()

Obtain either a “demo” or “paid” license from the Product Activation Server over the Internet node-locked licenses only!

[C/C++]
```
int retVal;
char icode[512];
retVal = sgPaGetInstallationCode(SG_IDTYPE_ETHERNET,icode);
```

[Objective C]
```
NSInteger retVal;
NSString *icode;
retVal = [PSProlibProxy sgPaGetInstallationCode:self forIdType:SG_IDTYPE_OS getInstallationCode:&icode];
```

[Java]
```
int retVal;
byte icode[] = new byte[512];
retVal = PSProlibProxy.sgPaGetInstallationCode(SG_IDTYPE.SG_IDTYPE_MACINTOSH,icode);
```

[Python]
```
retVal, icode = PSProlibProxy.sgPaGetInstallationCode(PSProlibProxy.SG_IDTYPE_OS)
```

[Visual Basic .NET]
```
Dim retVal As Integer
Dim icode As New StringBuilder(512)
retVal = PSProlibProxy.sgPaGetInstallationCode(SG_IDTYPE.SG_IDTYPE_VSN,icode)
```

[PHP 5]
```
$icode = str_repeat(' ',512);
$retVal = sgPaGetInstallationCode(SG_IDTYPE_MACINTOSH,$icode);
```

[MT5]
```
int retVal;
char icode[512];
retVal = sgPaGetInstallationCode(SG_IDTYPE_ETHERNET,icode);
```

[FORTRAN]
```
integer retVal
character*512 icode
retVal = sgPaGetInstallationCode(SG_IDTYPE_ETHERNET,icode)
```

Parameters

idtype [in]

The **SG_IDTYPE** desired. Can be one of the following. Not all are supported on each architecture

SG_IDTYPE_DEFAULT	**SG_IDTYPE_CURRENT**	**SG_IDTYPE_IPHONE**
SG_IDTYPE_HOSTID	**SG_IDTYPE_MACINTOSH**	**SG_IDTYPE_ANDROID**
SG_IDTYPE_BLACKBERRY	**SG_IDTYPE_ETHERNET**	**SG_IDTYPE_HOSTNAME**
SG_IDTYPE_IPADDRESS	**SG_IDTYPE_USERNAME**	**SG_IDTYPE_COUNTRY**
SG_IDTYPE_INET6	**SG_IDTYPE_VSN**	**SG_IDTYPE_VENDOR**
SG_IDTYPE_ANY	**SG_IDTYPE_COMPOSITE**	**SG_IDTYPE_OS**

icode [out]

The host ID in dashed format

Description

Used to obtain the *SafeGuard LM* host ID in a dashed format to be used with **sgPaGetDemoActivationKeys()** or **sgPaGetPaidActivationKeys()**. These series of functions are for node-locked licenses **only**.

Return codes

On success, **SG_SUCCESS** is returned.

On error, possible return codes are:

SG_IDTYPE_NOT_SUPPORTED
SG_CANT_GET_ID

sgPaGetDemoActivationKeys()

Used to obtain alternate style demo node-locked license keys from the Product Activation Server

```
[C/C++]
int retVal;
char ic[512],lk[128],ak[256];
retVal = sgPaGetDemoActivationKeys(“www.yourcompany.com”,29775,”f1”,”4.0”,NULL,ic,lk,ak);
```

```
[Objective C]
NSInteger retVal;
NSString *serverline,*componentline;
retVal = [PSProlibProxy sgPaGetDemoActivationKeys:self forWeb:@"98.220.92.50" forPort:29775
                                forComponent:@"monkey" forVersion:@"5.0" forEmail:@"user@gmail.com"
                                forInstallationCode:ic getLicenseKey:&lk getAuthorizationKey:&ak];
```

```
[Java]
int retVal;
byte ic[] = new byte[512];
byte lk[] = new byte[128];
byte ak[] = new byte[256];
retVal = PSProlibProxy.sgPaGetDemoActivationKeys(“www.yourcompany.com”,29775,”f1”,”4.0”,NULL,ic,lk,ak);
```

```
[Python]
retVal, ic, lk, ak = PSProlibProxy.sgPaGetDemoActivationKeys(“www.yourcompany.com”,29775,”f1”,”4.0”,NULL)
```

```
[Visual Basic .NET]
Dim retVal As Integer
Dim ic As New StringBuilder(512)
Dim lk As New StringBuilder(128)
Dim lk As New StringBuilder(256)
retVal = PSProlibProxy.sgPaGetDemoActivationKeys(“www.yourcompany.com”,29775,”f1”,”4.0”,NULL,ic,lk,ak)
```

```
[PHP 5]
$ic = str_repeat(' ',512);
$lk = str_repeat(' ',128);
$ak = str_repeat(' ',256);
$retVal = sgPaGetDemoActivationKeys('www.yourcompany.com',29775,'f1','4.0','userid@mail.com',$ic,$lk,$ak);
```

```
[MT5]
int retVal;
char web[256],component[16],version[12];
char ic[512],lk[128],ak[256];
StringToCharArray("www.yourcompany.com",web,0,WHOLE_ARRAY,CP_UTF8);
StringToCharArray("trademaster",component,0,WHOLE_ARRAY,CP_UTF8);
StringToCharArray("5.0",version,0,WHOLE_ARRAY,CP_UTF8);
retVal = sgPaGetDemoActivationKeys(web,29775,component,version,NULL,ic,lk,ak);
```

```
[FORTRAN]
integer retVal
character*512
character*256 ak
character*128 lk
retVal = sgPaGetDemoActivationKeys('www.yourcompany.com',29775,'f1','4.0','',ic,lk,ak)
```

Parameters

web [in]

The IP address or name of the activation server. i.e. **www.yourcompany.com**

port [in]

The TCP/IP port number the activation server or **sgproxyserver** is listening on

component [in]

The desired component name

version [in]

The desired version number

email [in]

The email address of the customer. This can actually be anything you wish to send back to the Product Activation Server

ic [out]

Installation code (host ID in XXX-YYYY-ZZZZ format)

lk [out]

License key (component, version, idtype and timeout in XXX-YYYY-ZZZZ format)

ak [out]

Authorization key (certificate in XXX-YYYY-ZZZZ format)

Description

This function sits on top of **sgPaActivate()** in such a manner that it treats a SafeGuard component line as three series of encrypted codes. It is designed to obfuscate the standard component line to make it more transparent to the end user, and sometimes easier to describe for Windows users.

Used to obtain a **demo** node-locked license from the Product Activation Server over the Internet. The Product Activation Server must be running and accessible and configured for the requested license to be successfully retrieved. Refer to the Product Activation Server documentation on how to configure and run. This function call will retrieve the necessary component line for your application to run. It must be properly installed once retrieved.

Return codes

On success, **SG_SUCCESS** is returned.

On error, possible return codes are:

SG_INVALIDHOST

SG_SERVERNORESPOND
SG_BADWRITE
SG_LOSTCONNECT
SG_BADREAD
SG_BADCOMMUNICATION
SG_BADCONNECT
SG_TIMEDIFF_TOO_LARGE
SG_COMPONENT_NOT_AVAILABLE
SG_COMPONENT_ALREADY_ISSUED
SG_CANT_QUERY_SQLDB
SG_CANT_CONNECT_TO_SQLDB
SG_WRONG_SERVER
SG_REJECTED
SG_COMPONENT_NAME_TOO_LONG
SG_VERSION_LENGTH_TOO_LONG

sgPaGetPaidActivationKeys()

Used to obtain alternate style paid node-locked license keys from the Product Activation Server

```
[C/C++]
int retVal;
char ic[512],lk[128],ak[256];
retVal = sgPaGetPaidActivationKeys(“www.yourcompany.com”,29775,NULL,”f1”,”4.0”,NULL,ic,lk,ak);
```

```
[Objective C]
NSInteger retVal;
NSString *serverline,*componentline;
retVal = [PSProlibProxy sgPaGetPaidActivationKeys:self forWeb:@"98.220.90.50" forPort:29775 forCust:NULL
                        forComponent:@"monkey" forVersion:@"5.0" forEmail:@"user@gmail.com"
                        forInstallationCode:ic getLicenseKey:&lk getAuthorizationKey:&ak];
```

```
[Java]
int retVal;
byte ic[] = new byte[512];
byte lk[] = new byte[128];
byte ak[] = new byte[256];
retVal =
PSProlibProxy.sgPaGetPaidActivationKeys(“www.yourcompany.com”,29775,NULL,”f1”,”4.0”,NULL,ic,lk,ak);
```

```
[Python]
retVal, ic, lk, ak =
PSProlibProxy.sgPaGetPaidActivationKeys(“www.yourcompany.com”,29775,NULL,”f1”,”4.0”,NULL)
```

```
[Visual Basic .NET]
Dim retVal As Integer
Dim ic As New StringBuilder(512)
Dim lk As New StringBuilder(128)
Dim lk As New StringBuilder(256)
retVal = PSProlibProxy.sgPaGetPaidActivationKeys(“www.yourcompany.com”,29775,
                        NULL,”f1”,”4.0”,NULL,ic,lk,ak)
```

```
[PHP 5]
$ic = str_repeat(' ',512);
$lk = str_repeat(' ',128);
$ak = str_repeat(' ',256);
$retVal = sgPaGetPaidActivationKeys('www.yourcompany.com',29775,'','f1','4.0','userid@mail.com',$ic,$lk,$ak);
```

```
[MT5]
int retVal;
char web[256],component[16],version[12],email[256];
char ic[512],lk[128],ak[256];
StringToCharArray("www.yourcompany.com",web,0,WHOLE_ARRAY,CP_UTF8);
StringToCharArray("trademaster",component,0,WHOLE_ARRAY,CP_UTF8);
StringToCharArray("5.0",version,0,WHOLE_ARRAY,CP_UTF8);
StringToCharArray("user@example.com",email,0,WHOLE_ARRAY,CP_UTF8);
retVal = sgPaGetDemoActivationKeys(web,29775,component,version,email,ic,lk,ak);
```

```
[FORTRAN]
integer retVal
character*512 ic
character*256 ak
```

```
character*128 lk
retVal = sgPaGetPaidActivationKeys('www.yourcompany.com',29775,NULL,'f1','4.0','',ic,lk,ak)
```

Parameters

web [in]

The IP address or name of the activation server. i.e. **www.yourcompany.com**

port [in]

The TCP/IP port number the activation server or **sgproxyserver** is listening on

cust [in]

The paid customer number. If **NULL**, then the Ethernet (MAC) address is used as the customer number

component [in]

The desired component name

version [in]

The desired version number

email [in]

The email address of the customer. This can actually be anything you wish to send back to the Product Activation Server

ic [out]

Installation code (host ID in XXX-YYYY-ZZZZ format)

lk [out]

License key (component, version, idtype and timeout in XXX-YYYY-ZZZZ format)

ak [out]

Authorization key (certificate in XXX-YYYY-ZZZZ format)

Description

This function sits on top of **sgPaActivate()** in such a manner that it treats a SafeGuard component line as three series of encrypted codes. It is designed to obfuscate the standard component line to make it more transparent to the end user, and sometimes easier to describe for Windows users.

Used to obtain a **paid** node-locked license from the Product Activation Server over the Internet. The Product Activation Server must be running and accessible and configured for the requested license to be successfully retrieved. Refer to the Product Activation Server documentation on how to configure and run. This function call will retrieve the necessary component line for your application to run. It must be properly installed once retrieved.

Return codes

On success, **SG_SUCCESS** is returned.

On error, possible return codes are:

SG_INVALIDHOST
SG_SERVERNORESPOND
SG_BADWRITE
SG_LOSTCONNECT
SG_BADREAD
SG_BADCOMMUNICATION
SG_BADCONNECT
SG_TIMEDIFF_TOO_LARGE
SG_COMPONENT_NOT_AVAILABLE
SG_COMPONENT_ALREADY_ISSUED
SG_CANT_QUERY_SQLDB
SG_CANT_CONNECT_TO_SQLDB
SG_WRONG_SERVER
SG_REJECTED
SG_COMPONENT_NAME_TOO_LONG
SG_VERSION_LENGTH_TOO_LONG

sgPaAssembleKeys()

Used to assemble the three license keys back into a *SafeGuard LM* component line.

```
[C/C++]
int retVal;
char ic[512],lk[128],ak[256],componentline[1024];
retVal = sgPaAssembleKeys(ic,lk,ak,componentline);
```

```
[Objective C]
NSInteger retVal;
NSString *componentline;
retVal = [PSProlibProxy sgPaAssembleKeys:self forInstallationCode:@"DB8E-D2EB-74D7-F867-BFC4-1C"
                                        forLicenseKey:@”C1D9-84B1-25D8-B020-A184-0DAA-AEFF-EE79-841B-E360"
                                        forAuthorizationKey:@"0ED6-4391-335C-F2D0-0980-7A6D-D217"
                                        getComponentLine:&componentline]
```

```
[Java]
int retVal;
byte ic[] = new byte[512];
byte lk[] = new byte[128];
byte ak[] = new byte[256];
byte componentline[] = new byte[1024];
retVal = PSProlibProxy.sgPaAssembleKeys(ic,lk,ak,componentline);
```

```
[Python]
retVal, componentline = PSProlibProxy.sgPaAssembleKeys("DB8E-D2EB-74D7-F867-BFC4-1C",
        ”C1D9-84B1-25D8-B020-A184-0DAA-AEFF-EE79-841B-E360","0ED6-4391-335C-F2D0-0980-7A6D-D217")
```

```
[Visual Basic .NET]
Dim retVal As Integer
Dim ic As New StringBuilder(512)
Dim lk As New StringBuilder(128)
Dim lk As New StringBuilder(256)
Dim componentline As New StringBuilder(1024)
retVal = PSProlibProxy.sgPaAssembleKeys(ic,lk,ak,componentline)
```

```
[PHP 5]
$ic = str_repeat(' ',512);
$lk = str_repeat(' ',128);
$ak = str_repeat(' ',256);
$componentline = str_repeat(' ',1024);
$retVal = sgPaAssembleKeys($ic,$lk,$ak,$componentline);
```

```
[MT5]
int retVal;
char ic[512],lk[128],ak[256],componentline[1024];
retVal = sgPaAssembleKeys(ic,lk,ak,componentline);
```

```
[FORTRAN]
integer retVal
character*1024 componentline
character*512  ic
character*256 ak
character*128 lk
retVal = sgPaAssembleKeys(ic,lk,ak,componentline)
```

Parameters

ic [in]

Installation code (host ID in XXX-YYYY-ZZZZ format)

lk [in]

License key (component, version, idtype and timeout in XXX-YYYY-ZZZZ format)

ak [in]

Authorization key (certificate in XXX-YYYY-ZZZZ format)

componentline [out]

The resulting component line

Description

Used to assemble the three keys back into a component line.

Return codes

On success, **SG_SUCCESS** is returned.

On error, possible return codes are:

SG_INVALID_CERTIFICATE

sgPaDecodeKey()

Used to decode the license key into the version, ID type and expiration date strings.

```
[C/C++]
int retVal;
char lk[128],component[16],version[12],idtype[16],expires[12];
retVal = sgPaDecodeKey(lk,component,version,idtype,expires);
```

```
[Objective C]
NSInteger retVal;
NSString *component,*version,*idtype,*expires;
retVal = [PSProlibProxy sgPaDecodeKey:self
                    forLicenseKey:@"C1D9-84B1-25D8-B020-A184-0DAA-AEFF-EE79-841B-E360"
                    getComponent:&component
                    getVersion:&version
                    getIdType:&idtype
                    getExpires:&expires]
```

```
[Java]
int retVal;
byte component[] = new byte[16];
byte version[] = new byte[12];
byte idtype[] = new byte[16];
byte expires[] = new byte[12];
retVal = PSProlibProxy.sgPaDecodeKey(lk,component,version,idtype,expires);
```

```
[Python]
retVal, component,version,idtype,expires = PSProlibProxy.sgPaDecodeKey("C1D9-84B1-25D8-B020-A184-0DAA-
AEFF-EE79-841B-E360")
```

```
[Visual Basic .NET]
Dim retVal As Integer
Dim lk As New StringBuilder(128)
Dim component As New StringBuilder(16)
Dim version As New StringBuilder(12)
Dim idtype As New StringBuilder(16)
Dim expires As New StringBuilder(12)
retVal = PSProlibProxy.sgPaDecodeKey(lk,component,version,idtype,expires)
```

```
[PHP]
$component = str_repeat(' ',16);
$version = str_repeat(' ',12);
$idtype = str_repeat(' ',16);
$expires = str_repeat(' ',12);
$retVal = sgPaDecodeKey($lk,$component,$version,$idtype,$expires);
```

```
[MT5]
int retVal;
char lk[128],component[16],version[12],idtype[16],expires[12];
retVal = sgPaDecodeKey(lk,component,version,idtype,expires);
```

```
[FORTRAN]
integer retVal
character*128 lk
character*16 component,idtype
```

```
character*12 version,expires
retVal = sgPaDecodeKey(lk,component,version,idtype,expires)
```

Parameters

lk [in]

License key (component, version, ID type and timeout in XXX-YYYY-ZZZZ format)

component [out]

The component name

version [out]

The version number of the component

idtype [out]

The ID type string

expires [out]

The expiration date. A date of “**never”** means the component does not expire

Description

Used to assemble the three keys back into a component line.

Return codes

On success, **SG_SUCCESS** is returned.

On error, possible return codes are:

SG_INVALID_CERTIFICATE

Node-locked license file manipulation routines:

sgExtractComponent()

Retrieves the next licensed component from the specified license file (or application for licenses stored inside the binary).

[C/C++]
```
int retVal;
char component[16],version[12],hostid[256],idtype[16],expires[12],certificate[128];
retVal = sgExtractComponent("@mylicense.lic",component,version,hostid,idtype,expires,certificate);
```

[Objective C]
```
NSInteger retVal;
NSString *component,*version,*hostid,*idtype,*expires,*certificate;
retVal = [PSProlibProxy sgExtractComponent:self forTarget:@"@mylicense.lic" getComponent:&component
            getVersion:&version getId:&hostid getIdType:&idtype getExpires:&expires getCertificate:&certificate];
```

[Java]
```
int retVal;
byte component[] = new byte[16];
byte version[] = new byte[12];
byte hostid[] = new byte[256];
byte idtype[] = new byte[16];
byte expires[] = new byte[12];
byte certificate[] = new byte[128];
retVal =
PSProlibProxy.sgExtractComponent("@mylicense.lic",component,version,hostid,idtype,expires,certificate);
```

[Python]
```
retVal, component, version, hostid, idtype, expires, certificate =
PSProlibProxy.sgExtractComponent("@mylicense.lic")
```

[Visual Basic .NET]
```
Dim retVal As Integer
Dim target As New String("@mylicense.lic")
Dim component As New StringBuilder(16)
Dim version As New StringBuilder(12)
Dim hostid As New StringBuilder(256)
Dim idtype As New StringBuilder(16)
Dim expires As New StringBuilder(12)
Dim certificate As New StringBuilder(128)
retVal =  PSProlibProxy.sgExtractComponent(target, component, version, hostid, idtype, expires, certificate)
```

[PHP 5]
```
$component = str_repeat(' ',16);
$version = str_repeat(' ',12);
$hostid = str_repeat(' ',256);
$idtype = str_repeat(' ',16);
$expires = str_repeat(' ',12);
$certificate = str_repeat(' ',128);
$retVal = sgExtractComponent('$HOME/mylicense.lic',$component,$version,$hostid,$idtype,$expires,$certificate);
```

[MT5]
```
int retVal;
char component[16],version[12],hostid[256],idtype[16],expires[12],certificate[128],licensefile[256];
StringToCharArray("@mylicense.lic",licensefile,0,WHOLE_ARRAY,CP_UTF8);
retVal = sgExtractComponent(licensefile,component,version,hostid,idtype,expires,certificate);
```

```
[FORTRAN]
integer retVal
character*256 hostid
character*128 certificate
character*16 component,idtype
character*12 version,expires
retVal = sgExtractComponent('@mylicense.lic',component,version,hostid,idtype,expires,certificate)
```

Parameters

target [in]

The license file path for **SG_LICENSETYPE_EXTERNAL** licenses. The application path for **SG_LICENSETYPE_INTERNAL** licenses. For **SG_LICENSETYPE_EXTERNAL** licenses, you can specify "@mylicense.lic" for example. The "@" tells the software the license file resides next to the application executable. This makes it easy to locate a license file without having to construct a pathname. You may also use environment variables and relative pathnames here too for Windows, UNIX and Linux

component [out]

The component name

version [out]

The version number of the component

hostid [out]

The hostid of the machine

idtype [out]

The idtype string

expires [out]

The expiration date. A date of "**never"** means the component does not expire

certificate [out]

The component line certificate which can be in either **RC4** or **SHA256** format

Description

Retrieves the next licensed component from the specified target. This functions purpose is to retrieve component licenses for display purposes. For example, you would use this function to retrieve the users licenses and display them however appropriate. This function can be called until **SG_NO_MORE_COMPONENTS_IN_FILE** is returned. This call is not valid when **SG_LICENSETYPE_FLOATING** is set.

Return codes

On success, **SG_SUCCESS** is returned.

On error, possible return codes are:

SG_NO_MORE_COMPONENTS_IN_FILE
SG_INVALID_CERTIFICATE
SG_TARGET_FILE_NOT_FOUND
SG_CANT_READ_TARGET_FILE
SG_INVALID_OPTION_FOR_LICENSETYPE

sgEnableComponent()

Adds the license for the specified component to the target license file (or application for licenses stored inside the binary).

[C/C++]
```
int retVal;
char component[16],version[12],hostid[256],idtype[16],expires[12],certificate[128];
retVal = sgEnableComponent(“@mylicense.lic”,component,version,hostid,idtype,expires,certificate);
```

[Objective C]
```
NSInteger retVal;
NSString *component,*version,*hostid,*idtype,*expires,*certificate;
retVal = [PSProlibProxy sgEnableComponent:self forTarget:@”@mylicense.lic” forComponent:&component
            forVersion:&version forId:&hostid forIdType:&idtype forExpires:&expires forCertificate:&certificate];
```

[Java]
```
int retVal;
byte component[] = new byte[16];
byte version[] = new byte[12];
byte hostid[] = new byte[256];
byte idtype[] = new byte[16];
byte expires[] = new byte[12];
byte certificate[] = new byte[128];
retVal =
PSProlibProxy.sgEnableComponent(“@mylicense.lic”,component,version,hostid,idtype,expires,certificate);
```

[Python]
```
retVal = PSProlibProxy.sgEnableComponent(“@mylicense.lic”,component, version, hostid, idtype, expires,
certificate)
```

[Visual Basic .NET]
```
Dim retVal As Integer
Dim target As New String("@mylicense.lic")
Dim component As New StringBuilder(16)
Dim version As New StringBuilder(12)
Dim hostid As New StringBuilder(256)
Dim idtype As New StringBuilder(16)
Dim expires As New StringBuilder(12)
Dim certificate As New StringBuilder(128)
retVal =  PSProlibProxy.sgEnableComponent(target, component, version, hostid, idtype, expires, certificate)
```

[PHP 5]
```
$component = str_repeat(' ',16);
$version = str_repeat(' ',12);
$hostid = str_repeat(' ',256);
$idtype = str_repeat(' ',16);
$expires = str_repeat(' ',12);
$certificate = str_repeat(' ',128);
$retVal = sgEnableComponent('$HOME/mylicense.lic',$component,$version,$hostid,$idtype,$expires,$certificate);
```

[MT5]
```
int retVal;
char component[16],version[12],hostid[256],idtype[16],expires[12],certificate[128],licensefile[256];
StringToCharArray("@mylicense.lic",licensefile,0,WHOLE_ARRAY,CP_UTF8);
retVal = sgEnableComponent(licensefile,component,version,hostid,idtype,expires,certificate);
```

```
[FORTRAN]
integer retVal
character*256 hostid
character*128 certificate
character*16 component, idtype
character*12 version, expires
retVal = sgEnableComponent('@mylicense.lic',component,version,hostid,idtype,expires,certificate)
```

Parameters

target [out]

The license file path for **SG_LICENSETYPE_EXTERNAL** licenses. The application path for **SG_LICENSETYPE_INTERNAL** licenses. For **SG_LICENSETYPE_EXTERNAL** licenses, you can specify “@mylicense.lic” for example. The “@” tells the software the license file resides next to the application executable. This makes it easy to locate a license file without having to construct a pathname. You may also use environment variables and relative pathnames here too for Windows, UNIX and Linux

component [in]

The component name

version [in]

The version number of the component

hostid [in]

The hostid of the machine

idtype [in]

The idtype string

expires [in]

The expiration date. A date of “**never”** means the component does not expire

certificate [in]

The component line certificate which can be in either **RC4** or **SHA256** format

Description

Enables or adds the license for the specified component. Typical use of this function is during delivery of the license to the user so that it can be permanently enabled for the application which needs the license. This call is not valid when **SG_LICENSETYPE_FLOATING** is set.

Return codes

On success, **SG_SUCCESS** is returned.

On error, possible return codes are:

SG_INVALID_TIMEOUT
SG_TARGET_FILE_NOT_FOUND
SG_CANT_CREATE_TARGET_FILE
SG_CANT_READ_TARGET_FILE
SG_CANT_WRITE_TO_TARGET_FILE
SG_COMPONENT_NAME_TOO_LONG
SG_VERSION_LENGTH_TOO_LONG
SG_VERSION_NOT_NUMERIC
SG_INVALID_CERTIFICATE
SG_NO_FREE_SLOTS
SG_INVALID_OPTION_FOR_LICENSETYPE

sgDisableComponent()

Removes the license for the specified component from the target license file (or application for licenses stored inside the binary).

[C/C++]
```
int retVal;
char component[16],version[12],hostid[256],idtype[16];
retVal = sgDisableComponent("@mylicense.lic",component,version,hostid,idtype);
```

[Objective C]
```
NSInteger retVal;
NSString *component,*version,*hostid,*idtype;
retVal = [PSProlibProxy sgDisableComponent:self forTarget:@"@mylicense.lic" forComponent:&component
         forVersion:&version forId:&hostid forIdType:&idtype];
```

[Java]
```
int retVal;
byte component[] = new byte[16];
byte version[] = new byte[12];
byte hostid[] = new byte[256];
byte idtype[] = new byte[16];
retVal = PSProlibProxy.sgDisableComponent("@mylicense.lic",component,version,hostid,idtype);
```

[Python]
```
retVal = PSProlibProxy.sgDisableComponent("@mylicense.lic",component, version, hostid, idtype)
```

[Visual Basic .NET]
```
Dim retVal As Integer
Dim target As New String("@mylicense.lic")
Dim component As New StringBuilder(16)
Dim version As New StringBuilder(12)
Dim hostid As New StringBuilder(256)
Dim idtype As New StringBuilder(16)
retVal =  PSProlibProxy.sgDisableComponent(target, component, version, hostid, idtype)
```

[PHP]
```
$component = str_repeat(' ',16);
$version = str_repeat(' ',12);
$hostid = str_repeat(' ',256);
$idtype = str_repeat(' ',16);
$retVal = sgDisableComponent('$HOME/mylicense.lic',$component,$version,$hostid,$idtype);
```

[MT5]
```
int retVal;
char component[16],version[12],hostid[256],idtype[16],licensefile[256];
StringToCharArray("@mylicense.lic",licensefile,0,WHOLE_ARRAY,CP_UTF8);
retVal = sgDisableComponent(licensefile,component,version,hostid,idtype,expires,certificate);
```

[FORTRAN]
```
integer retVal
character*256 hostid
character*16  component,idtype
character*12 version
retVal = sgDisableComponent('@mylicense.lic',component,version,hostid,idtype)
```

Parameters

target [in]

The license file path for **SG_LICENSETYPE_EXTERNAL** licenses. The application path for **SG_LICENSETYPE_INTERNAL** licenses. For **SG_LICENSETYPE_EXTERNAL** licenses, you can specify "@mylicense.lic" for example. The "@" tells the software the license file resides next to the application executable. This makes it easy to locate a license file without having to construct a pathname. You may also use environment variables and relative pathnames here too for Windows, UNIX and Linux

component [in]

The component name

version [in]

The version number of the component

hostid [in]

The hostid of the machine

idtype [in]

The idtype string

Description

Removes the license for the specified component. This call is not valid when **SG_LICENSETYPE_FLOATING** is set.

Return codes

On success, **SG_SUCCESS** is returned.

On error, possible return codes are:

SG_INVALID_TIMEOUT
SG_TARGET_FILE_NOT_FOUND
SG_CANT_CREATE_TARGET_FILE
SG_CANT_READ_TARGET_FILE
SG_CANT_WRITE_TO_TARGET_FILE
SG_COMPONENT_NAME_TOO_LONG
SG_VERSION_LENGTH_TOO_LONG
SG_VERSION_NOT_NUMERIC
SG_INVALID_CERTIFICATE
SG_NO_FREE_SLOTS
SG_INVALID_OPTION_FOR_LICENSETYPE

sgValidateComponent()

Given all the pieces of the component license, this function will verify they are valid.

```
[C/C++]
int retVal;
char component[16],version[12],hostid[256],idtype[16],expires[12],certificate[128];
retVal = sgValidateComponent(component,version,hostid,idtype,expires,certificate);
```

```
[Objective C]
NSInteger retVal;
NSString *component,*version,*hostid,*idtype,*expires,*certificate;
retVal = [PSProlibProxy sgValidateComponent:self forComponent:&component forVersion:&version
              forId:&hostid forIdType:&idtype forExpires:&expires forCertificate:&certificate];
```

```
[Java]
int retVal;
byte component[] = new byte[16];
byte version[] = new byte[12];
byte hostid[] = new byte[256];
byte idtype[] = new byte[16];
byte expires[] = new byte[12];
byte certificate[] = new byte[128];
retVal = PSProlibProxy.sgValidateComponent(component,version,hostid,idtype,expires,certificate);
```

```
[Python]
retVal= PSProlibProxy.sgValidateComponent(component, version, hostid, idtype, expires, certificate)
```

```
[Visual Basic .NET]
Dim retVal As Integer
Dim component As New StringBuilder(16)
Dim version As New StringBuilder(12)
Dim hostid As New StringBuilder(256)
Dim idtype As New StringBuilder(16)
Dim expires As New StringBuilder(12)
Dim certificate As New StringBuilder(128)
retVal =  PSProlibProxy.sgValidateComponent(component, version, hostid, idtype, expires, certificate)
```

```
[PHP 5]
byte component = str_repeat(' ',16);
byte version = str_repeat(' ',12);
byte hostid = str_repeat(' ',256);
byte idtype = str_repeat(' ',16);
byte expires = str_repeat(' ',12);
byte certificate = str_repeat(' ',128);
$retVal = sgValidateComponent($component,$version,$hostid,$idtype,$expires,$certificate);
```

```
[MT5]
int retVal;
char component[16],version[12],hostid[256],idtype[16],expires[12],certificate[128];
retVal = sgValidateComponent(component,version,hostid,idtype,expires,certificate);
```

```
[FORTRAN]
integer retVal
character*256 hostid
character*128 certificate
```

```
character*16 component,idtype
character*12 version,expires
retVal = sgValidateComponent(component,version,hostid,idtype,expires,certificate)
```

Parameters

component [in]

The component name

version [in]

The version number of the component

hostid [in]

The hostid of the machine

idtype [in]

The idtype string

expires [in]

The expiration date. A date of “**never”** means the component does not expire

certificate [in]

The component line certificate which can be in either **RC4** or **SHA256** format

Description

Validates the license for the specified component pieces. Typical use of this function is during delivery of the license to the user so that it can be permanently enabled for the application which needs the license. This call is not valid when **SG_LICENSETYPE_FLOATING** is set.

Return codes

On success, **SG_SUCCESS** is returned.

On error, possible return codes are:

SG_INVALID_TIMEOUT
SG_COMPONENT_NAME_TOO_LONG
SG_VERSION_LENGTH_TOO_LONG
SG_VERSION_NOT_NUMERIC
SG_INVALID_CERTIFICATE
SG_INVALID_OPTION_FOR_LICENSETYPE

sgEnableAllComponents()

Add all the licenses to the target license file found in the input file (or application for licenses stored inside the binary).

```
[C/C++]
int retVal;
retVal = sgEnableAllComponents(“@mylicense.lic”,”infile.txt”);
```

```
[Objective C]
NSInteger retVal;
retVal = [PSProlibProxy sgEnableAllComponents:self forTarget:@”@mylicense.lic” forFile:@”infile.txt”];
```

```
[Java]
int retVal;
retVal = PSProlibProxy.sgEnableAllComponents(“@mylicense.lic”,”infile.txt”);
```

```
[Python]
retVal = PSProlibProxy.sgEnableAllComponents(“@mylicense.lic”,”infile.txt”)
```

```
[Visual Basic .NET]
Dim retVal As Integer
Dim target As New String("@mylicense.lic")
Dim file As New String("infile.txt")
retVal =  PSProlibProxy.sgEnableAllComponents(target, file)
```

```
[PHP 5]
$retVal = sgEnableAllComponents('$HOME/mylicense.lic','infile.txt');
```

```
[MT5]
int retVal;
char target[256],file[256];
StringToCharArray("@mylicense.lic",target,0,WHOLE_ARRAY,CP_UTF8);
StringToCharArray(“infile.txt",file,0,WHOLE_ARRAY,CP_UTF8);
retVal = sgEnableAllComponents(target,file);
```

```
[FORTRAN]
integer retVal
retVal = sgEnableAllComponents('@mylicense.lic','infile.txt')
```

Parameters

target [out]

The license file path for **SG_LICENSETYPE_EXTERNAL** licenses. The application path for **SG_LICENSETYPE_INTERNAL** licenses. For **SG_LICENSETYPE_EXTERNAL** licenses, you can specify “@mylicense.lic” for example. The “@” tells the software the license file resides next to the application executable. This makes it easy to locate a license file without having to construct a pathname. You may also use environment variables and relative pathnames here too for Windows, UNIX and Linux

file [in]

The input file

Description

Enables or adds all the licenses found in the specified input file. Typical use of this function is during delivery of the license to the user so that it can be permanently enabled for the application which needs the license. This call is not valid when **SG_LICENSETYPE_FLOATING** is set.

Return codes

On success, **SG_SUCCESS** is returned.

On error, possible return codes are:

SG_CANT_READ_TEXT_FILE
SG_NO_MORE_COMPONENTS_IN_FILE
SG_INVALID_CERTIFICATE
SG_INVALID_TIMEOUT
SG_COMPONENT_NAME_TOO_LONG
SG_VERSION_LENGTH_TOO_LONG
SG_VERSION_NOT_NUMERIC
SG_INVALID_OPTION_FOR_LICENSETYPE

sgEnableComponentLine()

Installs the whole component line into the target license file (or application for licenses stored inside the binary).

[C/C++]
```
int retVal;
char componentline[1024];
retVal = sgEnableComponentLine(“@mylicense.lic”,componentline);
```

[Objective C]
```
NSInteger retVal;
NSString *componentline;
retVal = [PSProlibProxy sgEnableComponentLine:self forTarget:@”@mylicense.lic”
forComponent:&componentline];
```

[Java]
```
int retVal;
byte componentline[] = new byte[1024];
retVal = PSProlibProxy.sgEnableComponentLine(“@mylicense.lic”,componentline);
```

[Python]
```
retVal = PSProlibProxy.sgEnableComponentLine(“@mylicense.lic”,componentline)
```

[Visual Basic .NET]
```
Dim retVal As Integer
Dim target As New String("@mylicense.lic")
Dim componentline As New StringBuilder(1024)
retVal =  PSProlibProxy.sgEnableComponentLine(target, componentline)
```

[PHP 5]
```
byte componentline = str_repeat(' ',1024);
$retVal = sgEnableComponentLine('$HOME/mylicense.lic',componentline);
```

[MT5]
```
int retVal;
char target[256],componentline[1024];
StringToCharArray("@mylicense.lic",target,0,WHOLE_ARRAY,CP_UTF8);
StringToCharArray("NAME=monkey VERS=5.0 ID=w8814vhr0p1 IDTYPE=macintosh EXPIRES=never
CERT=0ed64391335cf2d00b987d01bc78",componentline,0,WHOLE_ARRAY,CP_UTF8);
retVal = sgEnableComponentLine(target, componentline);
```

[FORTRAN]
```
integer retVal
character*1024 componentline
retVal = sgEnableComponentLine('@mylicense.lic',componentline)
```

Parameters

target [out]

The license file path for **SG_LICENSETYPE_EXTERNAL** licenses. The application path for **SG_LICENSETYPE_INTERNAL** licenses. For **SG_LICENSETYPE_EXTERNAL** licenses, you can specify “@mylicense.lic” for example. The “@” tells the software the license file resides next to the application executable. This makes it easy to locate a license file without having to construct a pathname. You may also use environment variables and relative pathnames here too for Windows, UNIX and Linux componentline [in]

The complete component line

Description

Enables or adds the component line to the specified target. Typical use of this function is during delivery of the license to the user so that it can be permanently enabled for the application which needs the license. This call is not valid when **SG_LICENSETYPE_FLOATING** is set.

Return codes

On success, **SG_SUCCESS** is returned.

On error, possible return codes are:

SG_INVALID_CERTIFICATE
SG_INVALID_TIMEOUT
SG_TARGET_FILE_NOT_FOUND
SG_CANT_CREATE_TARGET_FILE
SG_CANT_WRITE_TO_TARGET_FILE
SG_CANT_READ_TARGET_FILE
SG_COMPONENT_NAME_TOO_LONG
SG_VERSION_LENGTH_TOO_LONG
SG_VERSION_NOT_NUMERIC;
SG_INVALID_OPTION_FOR_LICENSETYPE

General utility routines:

sgGenerateDate()

Generates a properly formatted *SafeGuard LM* date format in days from today

```
[C/C++]
int retVal;
char sgdate[16];
retVal = sgGenerateDate(30,sgdate);
```

```
[Objective C]
NSInteger retVal
NSString *sgdate;
retVal = [PSProlibProxy sgGenerateDate:self forDays:30 getDate:&sgdate];
```

```
[Java]
int retVal.days;
byte sgdate[] = new byte[16];
retVal = PSProlibProxy.sgGenerateDate(30,sgdate);
```

```
[Python]
retVal, sgdate = PSProlibProxy.sgGenerateDate(30)
```

```
[Visual Basic .NET]
Dim retVal As Integer
Dim datestring As New StringBuilder(16)
retVal =  PSProlibProxy.sgGenerateDate(30,sgdate)
```

```
[PHP 5]
$sgdate = str_repeat(' ',16);
$retVal = sgGenerateDate(15,$sgdate);
```

```
[MT5]
int retVal;
char sgdate[16];
retVal = sgGenerateDate(15,sgdate);
```

```
[FORTRAN]
integer retVal
character*16 sgdate
retVal = sgGenerateDate(30,sgdate)
```

Parameters

days [in]

The number of days from now to generate the SafeGuard date

sgdate [out]

The generated date

Description

Generates a properly formatted *SafeGuard LM* date format in days from today. The valid range for days *i*s 0 to 9900. A value of -1 days returns the **never** timeout date.

Return codes

On success, **SG_SUCCESS** is returned.

SG_SUCCESS is the only return code for this function.

sgArc4()

Encode or decode a string using **RC4** encoding.

[C/C++]
```
int retVal;
char instring[16],outstring[32];
retVal = sgArc4(1,”Secret”,instring,outstring);
```

[Objective C]
```
NSInteger retVal
NSString *result;
retVal = [PSProlibProxy sgArc4:self forDirection:1 forKey:@"Secret"
                                            forString:@"The mouse ran up the clock!" getResult:&result];
```

[Java]
```
int retVal.days;
byte instring[] = new byte[16];
byte outstring[] = new byte[32];
retVal = PSProlibProxy.sgArc4(1,”Secret”,instring,outstring);
```

[Python]
```
retVal, outstring = PSProlibProxy.sgArc4(1,”Secret”,”Attack at dawn”)
```

[Visual Basic .NET]
```
Dim retVal As Integer
Dim instring As New StringBuilder(16)
Dim outstring As New StringBuilder(32)
retVal =  PSProlibProxy.sgArc4(1,”Secret”,instring,outstring)
```

[PHP 5]
```
$outstring = str_repeat(' ',32);
$retVal = sgArc4(1,'Secret','Attack at dawn',$oustring);
```

[MT5]
```
int retVal;
char crtpy[32],message[32],outstring[65];
StringToCharArray("Secret",crypt,0,WHOLE_ARRAY,CP_UTF8);
StringToCharArray("Attach at dawn",message,0,WHOLE_ARRAY,CP_UTF8);
retVal = sgArc5(1,crypt,message,outstring);
```

[FORTRAN]
```
character*32 outstring
retVal = sgArc4(1,'Secret','Attack at dawn',oustring)
```

Parameters

direction [in]

1 = encrypt, 0 = decrypt.

key [int]

The encryption/decryption key

instring [in]

The string to encode or decode

outstring [out]

The the result string

Description

Encode or decode a string using RC4 encoding. The output string must be twice the size of the input string when encoding and can be half the size when decoding.

Return codes

On success, **SG_SUCCESS** is returned.

SG_SUCCESS is the only return code for this function.