Overview

OnGuard is a library of components, classes, and routines that allow you to protect your applications after they are released to the public. Using OnGuard, you could release an application that is partially functional so that users can try it. When a user is ready to purchase the fully functional application, you supply a release code to unlock all of the features (or the subset that the user is purchasing). You can make your application readily available to a large number of potential users, but still protect your investment. Application protection is accomplished through the use of *keys* to lock or restrict one or more features of an application and several types of *release codes* (or access codes) to enable them.

Contents

OnGuard is a library of components, classes, and routines that allow you to protect your applications after they are released to the public. Using OnGuard, you could release an application that is partially functional so that users can try it. When a user is ready to purchase the fully functional application, you supply a release code to unlock all of the features (or the subset that the user is purchasing). You can make your application readily available to a large number of potential users, but still protect your investment. Application protection is accomplished through the use of *keys* to lock or restrict one or more features of an application and several types of *release codes* (or access codes) to enable them.

OgFirst Unit

Low-Level Routines

API Reference

License

Version 1.13 is the original source released by TurboPower.
Delphi 7 support was added in this release.
This release was ported to CLX.
The CLX port was then ported to FPC/Lazarus.
SongBeamer added packages for Delphi 2009 and Delphi 2010 and made some changes for Unicode support.

Version 1.14 was created by Roman Kassebaum.
This version only had packages for Delphi 2009 and Delphi 2010 with the new version number.
There were newsgroup postings saying it did not compile where the SongBeamer release did.

Version 1.15 was created by Andrew Haines.
Packages for Delphi XE through XE5 were added.
Source version numbers were updated.
A merge of the 1.13, 1.14, SongBeamer, CLX, and FPC/Lazarus ports was started.
Unit tests for a number of the API routines were created using Delphi XE5 and DUnit.
Unit test values were pulled from Delphi 6 running version 1.13.
The original HLP file was imported into a Help and Manual project.
The H&M project was exported to CHM and HxS files as well as HTML.
The help has been expanded to include the various types, files, and routines.
Screen shots have been added to the help file.
The SourceForge feature request 5 has been implemented.
The SourceForge bug reports 6, 7, 8, and 10 have been implemented.

Mozilla Public License Version 1.1

1. Definitions.

1.0.1. "Commercial Use"
means distribution or otherwise making the Covered Code available to a third party.

1.1. "Contributor"
means each entity that creates or contributes to the creation of Modifications.

1.2. "Contributor Version"
means the combination of the Original Code, prior Modifications used by a Contributor, and the Modifications made by that particular Contributor.

1.3. "Covered Code"
means the Original Code or Modifications or the combination of the Original Code and Modifications, in each case including portions thereof.

1.4. "Electronic Distribution Mechanism"
means a mechanism generally accepted in the software development community for the electronic transfer of data.

1.5. "Executable"
means Covered Code in any form other than Source Code.

1.6. "Initial Developer"
means the individual or entity identified as the Initial Developer in the Source Code notice required by Exhibit A.

1.7. "Larger Work"
means a work which combines Covered Code or portions thereof with code not governed by the terms of this License.

1.8. "License"

means this document.

1.8.1. "Licensable"

means having the right to grant, to the maximum extent possible, whether at the time of the initial grant or subsequently acquired, any and all of the rights conveyed herein.

1.9. "Modifications"

means any addition to or deletion from the substance or structure of either the Original Code or any previous Modifications. When Covered Code is released as a series of files, a Modification is:

Any addition to or deletion from the contents of a file containing Original Code or previous Modifications.

Any new file that contains any part of the Original Code or previous Modifications.

1.10. "Original Code"

means Source Code of computer software code which is described in the Source Code notice required by Exhibit A as Original Code, and which, at the time of its release under this License is not already Covered Code governed by this License.

1.10.1. "Patent Claims"

means any patent claim(s), now owned or hereafter acquired, including without limitation, method, process, and apparatus claims, in any patent Licensable by grantor.

1.11. "Source Code"

means the preferred form of the Covered Code for making modifications to it, including all modules it contains, plus any associated interface definition files, scripts used to control compilation and installation of an Executable, or source code differential comparisons against either the Original Code or another well known, available Covered Code of the Contributor's choice. The Source Code can be in a compressed or archival form, provided the appropriate

decompression or de-archiving software is widely available for no charge.

1.12. "You" (or "Your")

means an individual or a legal entity exercising rights under, and complying with all of the terms of, this License or a future version of this License issued under Section 6.1. For legal entities, "You" includes any entity which controls, is controlled by, or is under common control with You. For purposes of this definition, "control" means (a) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (b) ownership of more than fifty percent (50%) of the outstanding shares or beneficial ownership of such entity.

2. Source Code License.

2.1. The Initial Developer Grant.

The Initial Developer hereby grants You a world-wide, royalty-free, non-exclusive license, subject to third party intellectual property claims:

under intellectual property rights (other than patent or trademark) Licensable by Initial Developer to use, reproduce, modify, display, perform, sublicense and distribute the Original Code (or portions thereof) with or without Modifications, and/or as part of a Larger Work; and

under Patents Claims infringed by the making, using or selling of Original Code, to make, have made, use, practice, sell, and offer for sale, and/or otherwise dispose of the Original Code (or portions thereof).

the licenses granted in this Section 2.1 (a) and (b) are effective on the date Initial Developer first distributes Original Code under the terms of this License.

Notwithstanding Section 2.1 (b) above, no patent license is granted: 1) for code that You delete from the Original Code; 2) separate from the Original Code; or 3) for infringements caused by: i) the modification of the Original Code or ii) the combination of the Original Code with other software or devices.

2.2. Contributor Grant.

Subject to third party intellectual property claims, each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license

under intellectual property rights (other than patent or trademark) Licensable by Contributor, to use, reproduce, modify, display, perform, sublicense and distribute the Modifications created by such Contributor (or portions thereof) either on an unmodified basis, with other Modifications, as Covered Code and/or as part of a Larger Work; and

under Patent Claims infringed by the making, using, or selling of Modifications made by that Contributor either alone and/or in combination with its Contributor Version (or portions of such combination), to make, use, sell, offer for sale, have made, and/or otherwise dispose of: 1) Modifications

made by that Contributor (or portions thereof); and 2) the combination of Modifications made by that Contributor with its Contributor Version (or portions of such combination).

the licenses granted in Sections 2.2 (a) and 2.2 (b) are effective on the date Contributor first makes Commercial Use of the Covered Code.

Notwithstanding Section 2.2 (b) above, no patent license is granted: 1) for any code that Contributor has deleted from the Contributor Version; 2) separate from the Contributor Version; 3) for infringements caused by: i) third party modifications of Contributor Version or ii) the combination of Modifications made by that Contributor with other software (except as part of the Contributor Version) or other devices; or 4) under Patent Claims infringed by Covered Code in the absence of Modifications made by that Contributor.

3. Distribution Obligations.

3.1. Application of License.

The Modifications which You create or to which You contribute are governed by the terms of this License, including without limitation Section 2.2. The Source Code version of Covered Code may be distributed only under the terms of this License or a future version of this License released under Section 6.1, and You must include a copy of this License with every copy of the Source Code You distribute. You may not offer or impose any terms on any Source Code version that alters or restricts the applicable version of this License or the recipients' rights hereunder. However, You may include an additional document offering the additional rights described in Section 3.5.

3.2. Availability of Source Code.

Any Modification which You create or to which You contribute must be made available in Source Code form under the terms of this License either on the same media as an Executable version or via an accepted Electronic Distribution Mechanism to anyone to whom you made an Executable version available; and if made available via Electronic Distribution Mechanism, must remain available for at least twelve (12) months after the date it initially became available, or at least six (6) months after a subsequent version of that particular Modification has been made available to such recipients. You are responsible for ensuring that the Source Code version remains available even if the Electronic Distribution Mechanism is maintained by a third party.

3.3. Description of Modifications.

You must cause all Covered Code to which You contribute to contain a file documenting the changes You made to create that Covered Code and the date of any change. You must include a prominent statement that the Modification is derived, directly or indirectly, from Original Code provided by the Initial Developer and including the name of the Initial Developer in (a) the Source Code, and (b) in any notice in an Executable version or related documentation

in which You describe the origin or ownership of the Covered Code.

3.4. Intellectual Property Matters

(a) Third Party Claims

If Contributor has knowledge that a license under a third party's intellectual property rights is required to exercise the rights granted by such Contributor under Sections 2.1 or 2.2, Contributor must include a text file with the Source Code distribution titled "LEGAL" which describes the claim and the party making the claim in sufficient detail that a recipient will know whom to contact. If Contributor obtains such knowledge after the Modification is made available as described in Section 3.2, Contributor shall promptly modify the LEGAL file in all copies Contributor makes available thereafter and shall take other steps (such as notifying appropriate mailing lists or newsgroups) reasonably calculated to inform those who received the Covered Code that new knowledge has been obtained.

(b) Contributor APIs

If Contributor's Modifications include an application programming interface and Contributor has knowledge of patent licenses which are reasonably necessary to implement that API, Contributor must also include this information in the legal file.

(c) Representations.

Contributor represents that, except as disclosed pursuant to Section 3.4 (a) above, Contributor believes that Contributor's Modifications are Contributor's original creation(s) and/or Contributor has sufficient rights to grant the rights conveyed by this License.

3.5. Required Notices.

You must duplicate the notice in Exhibit A in each file of the Source Code. If it is not possible to put such notice in a particular Source Code file due to its structure, then You must include such notice in a location (such as a relevant directory) where a user would be likely to look for such a notice. If You created one or more Modification(s) You may add your name as a Contributor to the notice described in Exhibit A. You must also duplicate this License in any documentation for the Source Code where You describe recipients' rights

or ownership rights relating to Covered Code. You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations to one or more recipients of Covered Code. However, You may do so only on Your own behalf, and not on behalf of the Initial Developer or any Contributor. You must make it absolutely clear than any such warranty, support, indemnity or liability obligation is offered by You alone, and You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of warranty, support, indemnity or liability terms You offer.

3.6. Distribution of Executable Versions.

You may distribute Covered Code in Executable form only if the requirements of Sections 3.1, 3.2, 3.3, 3.4 and 3.5 have been met for that Covered Code, and if You include a notice stating that the Source Code version of the Covered Code is available under the terms of this License, including a description of how and where You have fulfilled the obligations of Section 3.2. The notice must be conspicuously included in any notice in an Executable version, related documentation or collateral in which You describe recipients' rights relating to the Covered Code. You may distribute the Executable version of Covered Code or ownership rights under a license of Your choice, which may contain terms different from this License, provided that You are in compliance with the terms of this License and that the license for the Executable version does not attempt to limit or alter the recipient's rights in the Source Code version from the rights set forth in this License. If You distribute the Executable version under a different license You must make it absolutely clear that any terms which differ from this License are offered by You alone, not by the Initial Developer or any Contributor. You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of any such terms You offer.

3.7. Larger Works.

You may create a Larger Work by combining Covered Code with other code not governed by the terms of this License and distribute the Larger Work as a single product. In such a case, You must make sure the requirements of this License are fulfilled for the Covered Code.

4. Inability to Comply Due to Statute or Regulation.

If it is impossible for You to comply with any of the terms of this License with respect to some or all of the Covered Code due to statute, judicial order, or regulation then You must: (a) comply with the terms of this License to the maximum extent possible; and (b) describe the limitations and the code they affect. Such description must be included in the legal file described in Section 3.4 and must be included with all distributions of the Source Code. Except to the extent prohibited by statute or regulation, such description must be sufficiently detailed for a recipient of ordinary skill to be able to understand it.

5. Application of this License.

This License applies to code to which the Initial Developer has attached the notice in Exhibit A and to related Covered Code.

6. Versions of the License.

6.1. New Versions

Netscape Communications Corporation ("Netscape") may publish revised and/or new versions of the License from time to time. Each version will be given a distinguishing version number.

6.2. Effect of New Versions

Once Covered Code has been published under a particular version of the License, You may always continue to use it under the terms of that version. You may also choose to use such Covered Code under the terms of any subsequent version of the License published by Netscape. No one other than Netscape has the right to modify the terms applicable to Covered Code created under this License.

6.3. Derivative Works

If You create or use a modified version of this License (which you may only do in order to apply it to code which is not already Covered Code governed by this License), You must (a) rename Your license so that the phrases "Mozilla", "MOZILLAPL", "MOZPL", "Netscape", "MPL", "NPL" or any confusingly similar phrase do not appear in your license (except to note that your license differs from this License) and (b) otherwise make it clear that Your version of the license contains terms which differ from the Mozilla Public License and Netscape Public License. (Filling in the name of the Initial Developer, Original Code or Contributor in the notice described in Exhibit A shall not of themselves be deemed to be modifications of this License.)

7. Disclaimer of warranty

Covered code is provided under this license on an "as is" basis, without warranty of any kind, either expressed or implied, including, without limitation, warranties that the covered code is free of defects, merchantable, fit for a particular purpose or non-infringing. The entire risk as to the quality and performance of the covered code is with you. Should any covered code prove defective in any respect, you (not the initial developer or any other contributor) assume the cost of any necessary servicing, repair or correction. This disclaimer of warranty constitutes an essential part of this license. No use of any covered code is authorized hereunder except under this disclaimer.

8. Termination

8.1. This License and the rights granted hereunder will terminate automatically if You fail to comply with terms herein and fail to cure such breach within 30 days of becoming aware of the breach. All sublicenses to the Covered Code which are properly granted shall survive any termination of this License. Provisions which, by their nature, must remain in effect beyond the termination of this License shall survive.

8.2. If You initiate litigation by asserting a patent infringement claim (excluding declatory judgment actions) against Initial Developer or a Contributor (the Initial Developer or Contributor against whom You file such action is referred to as "Participant") alleging that:

such Participant's Contributor Version directly or indirectly infringes any patent, then any and all rights granted by such Participant to You under Sections 2.1 and/or 2.2 of this License shall, upon 60 days notice from Participant terminate prospectively, unless if within 60 days after receipt of notice You either: (i) agree in writing to pay Participant a mutually agreeable reasonable royalty for Your past and future use of Modifications made by such Participant, or (ii) withdraw Your litigation claim with respect to the Contributor Version against such Participant. If within 60 days of notice, a reasonable royalty and payment arrangement are not mutually agreed upon in writing by the parties or the litigation claim is not withdrawn, the rights granted by Participant to You under Sections 2.1 and/or 2.2 automatically terminate at the expiration of the 60 day notice period specified above.

any software, hardware, or device, other than such Participant's Contributor Version, directly or indirectly infringes any patent, then any rights granted to You by such Participant under Sections 2.1(b) and 2.2(b) are revoked effective as of the date You first made, used, sold, distributed, or had made, Modifications made by that Participant.

8.3. If You assert a patent infringement claim against Participant alleging that such Participant's Contributor Version directly or indirectly infringes any

patent where such claim is resolved (such as by license or settlement) prior to the initiation of patent infringement litigation, then the reasonable value of the licenses granted by such Participant under Sections 2.1 or 2.2 shall be taken into account in determining the amount or value of any payment or license.

8.4. In the event of termination under Sections 8.1 or 8.2 above, all end user license agreements (excluding distributors and resellers) which have been validly granted by You or any distributor hereunder prior to termination shall survive termination.

9. Limitation of liability

Under no circumstances and under no legal theory, whether tort (including negligence), contract, or otherwise, shall you, the initial developer, any other contributor, or any distributor of covered code, or any supplier of any of such parties, be liable to any person for any indirect, special, incidental, or consequential damages of any character including, without limitation, damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses, even if such party shall have been informed of the possibility of such damages. This limitation of liability shall not apply to liability for death or personal injury resulting from such party's negligence to the extent applicable law prohibits such limitation. Some jurisdictions do not allow the exclusion or limitation of incidental or consequential damages, so this exclusion and limitation may not apply to you.

10. U.S. government end users

The Covered Code is a "commercial item," as that term is defined in 48 C.F.R. 2.101 (Oct. 1995), consisting of "commercial computer software" and "commercial computer software documentation," as such terms are used in 48 C.F.R. 12.212 (Sept. 1995). Consistent with 48 C.F.R. 12.212 and 48 C.F.R. 227.7202-1 through 227.7202-4 (June 1995), all U.S. Government End Users acquire Covered Code with only those rights set forth herein.

11. Miscellaneous

This License represents the complete agreement concerning subject matter hereof. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. This License shall be governed by California law provisions (except to the extent applicable law, if any, provides otherwise), excluding its conflict-of-law provisions. With respect to disputes in which at least one party is a citizen of, or an entity chartered or registered to do business in the United States of America, any litigation relating to this License shall be subject to the jurisdiction of the Federal Courts of the Northern District of California, with venue lying in Santa Clara County, California, with the losing party responsible for costs, including without limitation, court costs and reasonable attorneys' fees and expenses. The application of the United Nations Convention on Contracts for the International Sale of Goods is expressly excluded. Any law or regulation which provides that the language of a contract shall be construed against the drafter shall not apply to this License.

12. Responsibility for claims

As between Initial Developer and the Contributors, each party is responsible for claims and damages arising, directly or indirectly, out of its utilization of rights under this License and You agree to work with Initial Developer and Contributors to distribute such responsibility on an equitable basis. Nothing herein is intended or shall be deemed to constitute any admission of liability.

13. Multiple-licensed code

Initial Developer may designate portions of the Covered Code as "Multiple-Licensed". "Multiple-Licensed" means that the Initial Developer permits you to utilize portions of the Covered Code under Your choice of the MPL or the alternative licenses, if any, specified by the Initial Developer in the file described in Exhibit A.

Exhibit A - Mozilla Public License.

"The contents of this file are subject to the Mozilla Public License
Version 1.1 (the "License"); you may not use this file except in
compliance with the License. You may obtain a copy of the License at
http://www.mozilla.org/MPL/

Software distributed under the License is distributed on an "AS IS"
basis, WITHOUT WARRANTY OF ANY KIND, either express or implied.
See the
License for the specific language governing rights and limitations
under the License.

The Original Code is _____.

The Initial Developer of the Original Code is _____.
Portions created by _____ are Copyright (C) _____
_____. All Rights Reserved.

Contributor(s): _____.

Alternatively, the contents of this file may be used under the terms
of the _____ license (the "[___] License"), in which case the
provisions of [_____] License are applicable instead of those
above. If you wish to allow use of your version of this file only
under the terms of the [____] License and not to allow others to use
your version of this file under the MPL, indicate your decision by
deleting the provisions above and replace them with the notice and
other provisions required by the [___] License. If you do not delete
the provisions above, a recipient may use your version of this file
under either the MPL or the [___] License."

NOTE: The text of this Exhibit A may differ slightly from the text of the
notices in the Source Code files of the Original Code. You should use the text
of this Exhibit A rather than the text found in the Original Code Source Code

for Your Modifications.

TOgCodeBase

The TOgCodeBase class is the ancestor class for the other "release code" components. It implements several properties and methods that are common for all of its descendants.

TOgMakeKeys
TOgMakeCodes

TOgDateCode
TOgDaysCode
TOgNetCode
TOgRegistrationCode
TOgSerialNumberCode
TOgSpecialCode
TOgUsageCode
TOgNetCode

TOgProtectExe

TOgDateCode implements a Start/End Date release code. Use this release code when you need to limit the amount of time that an application (or specific features of an application) can be used. Both a start date and an end date are encoded into this release code. This allows you to detect a change to the computers clock that results in a date outside of the date range or an attempt to alter the registry or INI file entry.

TOgDaysCode implements a Number of Days Used release code. This release code limits the number of days that an application (or specific features) can be used. The application can be run an unlimited number of times each day.

TOgMakeCodes is a non-visual component that displays a dialog when its Execute method is called. The dialog allows you to create several types of release codes. Each release code consists of 8 bytes and is viewed and entered as 16 hexadecimal digits.

TOgMakeKeys is a non-visual component that displays a dialog when its Execute method is called. The dialog allows you to create and maintain keys. Keys are used to encode and decode the release codes that the other OnGuard components use.

TOgNetCode implements a Network Metering release code. This release code limits the number of concurrent instances of an application that are allowed to run on a network. It does this through the use of a network release code and a Network Access File. The use of a network release code is no different than other release codes, but there are additional maintenance issues related to the network file that your application must handle.

The TOgProtectExe component allows you to detect changes to your EXE file. The size of the EXE file and a 32-bit CRC (Cyclical Redundancy Check) value are recorded in the EXE file and checked each time the application is run.

TOgRegistrationCode implements a Simple Registration release code. This release code ties the users name, company name, or some other textual data to the registration code.

TOgSerialNumberCode implements a Serial Number Registration release code. This release code ties a serial number to the release code. This release code is very similar to the Simple Registration release code. The only difference is in the data that is used as part of the code generation process. The Serial Number Registration release code uses a number instead of a text string.

TOgSpecialCode implements a Special Registration release code. This release code is based on a special value (a long integer) that can be used to indicate anything you like.

TOgUsageCode implements a Usage Count release code. This release code limits the number of times an application can be executed.

Generate Key Routines

- GenerateRandomKeyPrim
- GenerateMD5KeyPrim
- GenerateTMDKeyPrim

Modifier Routines

- ApplyModifierToKeyPrim
- CreateMachineID
- GenerateDateModifierPrim
- GenerateMachineModifierPrim
- GenerateStringModifierPrim
- GenerateUniqueModifierPrim

Hash Routines

- StringHashElf

Mixing Routines

- MixBlock

Utility Routines

- ExpandDate
- ShrinkDate
- BufferToHex
- BufferToHexBytes
- HexStringIsZero
- HexToBuffer
- GetCodeType
- GetExpirationDate

Date Code

- GetDateCodeValue
- InitDateCode
- IsDateCodeExpired
- IsDateCodeValid
- InitDateCodeEx
- GetDateCodeStart

- GetDateCodeEnd

Days Code

- DecDaysCode
- GetDaysCodeValue
- InitDaysCode
- IsDaysCodeExpired
- IsDaysCodeValid

Registration Code

- InitRegCode
- IsRegCodeExpired
- IsRegCodeValid
- IsRegCodeRegisteredTo

Serial Number Code

- GetSerialNumberCodeValue
- InitSerialNumberCode
- IsSerialNumberCodeExpired
- IsSerialNumberCodeValid
- IsSerialNumberCodeValidFor

Special Code

- GetSpecialCodeValue
- InitSpecialCode
- IsSpecialCodeExpired
- IsSpecialCodeValid
- IsSpecialCodeValidFor

Usage Code

- DecUsageCode
- GetUsageCodeValue
- InitUsageCode
- IsUsageCodeExpired
- IsUsageCodeValid
- InitUsageCodeUnlimited

Network Code

- IsAppOnNetwork
- CheckNetAccessFile
- CreateNetAccessFile
- CreateNetAccessFileEx
- DecodeNAFCountCode
- GetNetAccessFileInfo
- EncodeNAFCountCode
- LockNetAccessFile
- ResetNetAccessFile
- UnlockNetAccessFile

Protect EXE

- IsExeTampered
- ProtectExe
- UnprotectExe
- UpdateChecksum
- FileCRC32
- UpdateCRC32

Single Instance

- IsFirstInstance
- ActivateFirstInstance

Enter topic text here.

```
{$IFDEF Win16}
 DWord      = LongInt;
 PDWord     = ^DWord;
 TGUID      = GUID;   {Delphi 1.0 defines it as GUID - Delphi 2.0 defines it as
TGUID}
 AnsiChar   = Char;
 PAnsiChar  = PChar;
 {$ENDIF}

 {$IFNDEF FPC}
 PByte        = ^Byte;
 PByteArray    = ^TByteArray;
 TByteArray    = array [0..MaxStructSize div SizeOf(Byte) - 1] of Byte;
 PLongInt      = ^LongInt;
 {$ENDIF}
 PLongIntArray = ^TLongIntArray;
 TLongIntArray = array [0..MaxStructSize div SizeOf(LongInt) - 1] of
LongInt;
```

TLongIntRec
PCode
TCode
TCodeType
TKey
TKeyType
TTMDContext
TMD5Context
TMD5Digest
T128Bit
T256Bit
TEsMachineInfoSet
TCodeStatus

TNetAccess
TNetAccessInfo
TGetFileNameEvent

```
PSignatureRec = ^TSignatureRec;
TSignatureRec = packed record
  Sig1   : DWord;                                    {!!.07}
  Sig2   : DWord;                                    {!!.07}
  Sig3   : DWord;                                    {!!.07}
  Offset : DWord;                                    {!!.07}
  Size   : DWord;                                    {!!.07}
  CRC    : DWord;                                    {!!.07}
  Sig4   : DWord;                                    {!!.07}
  Sig5   : DWord;                                    {!!.07}
  Sig6   : DWord;                                    {!!.07}
end;

TExeStatus = (
  exeSuccess,          {no error}
  exeSizeError,        {the file size has changed}
  exeIntegrityError,   {CRC does not match}
  exeNotStamped,       {the exe has not been stamped}
  exeAccessDenied      {share violation}                    {!!.05}
);

TCheckedExeEvent = procedure(Sender : TObject; Status : TExeStatus) of
object;
```

Enter topic text here.

Enter topic text here.

Enter topic text here.

Types

```
{$IFDEF Win16}
 DWord      = LongInt;
 PDWord     = ^DWord;
 TGUID      = GUID;   {Delphi 1.0 defines it as GUID - Delphi 2.0 defines it as
TGUID}
 AnsiChar   = Char;
 PAnsiChar  = PChar;
 {$ENDIF}

 {$IFNDEF FPC}
 PByte        = ^Byte;
 PByteArray    = ^TByteArray;
 TByteArray    = array [0..MaxStructSize div SizeOf(Byte) - 1] of Byte;
 PLongInt      = ^LongInt;
 {$ENDIF}
 PLongIntArray = ^TLongIntArray;
 TLongIntArray = array [0..MaxStructSize div SizeOf(LongInt) - 1] of
LongInt;
```

TLongIntRec
PCode
TCode
TCodeType
TKey
TKeyType
TTMDContext
TMD5Context
TMD5Digest
T128Bit
T256Bit
TEsMachineInfoSet
TCodeStatus

Constants

```
DefAutoCheck      = True;
 DefAutoDecrease   = True;
 DefCheckSize      = True;
 DefStoreCode      = False;
 DefStoreModifier  = False;
 DefStoreRegString = False;

 OgVersionStr      = '1.15';

{magic values}
 DaysCheckCode    = Word($649B);
 DateCheckCode    = Word($A4CB);
 NetCheckCode     = Word($9341);
 RegCheckCode     = Word($D9F6);
 SerialCheckCode  = Word($3C69);
 UsageCheckCode   = Word($F3D5);
 SpecialCheckCode = Word($9C5B);

 {$IFDEF Win32}
 MaxStructSize = 1024 * 2000000; {2G}
 {$ELSE}
 MaxStructSize = 1024 * 64 - 1;  {64K}
 {$ENDIF}

 DefCodeType      = ctDate;
 DefKeyType       = ktRandom;
```

BaseDate

Exceptions

```
EOnGuardException = class(Exception);
 EOnGuardBadDateException = class(EOnGuardException);                {!!.15}
 EOnGuardClockIssueException = class(EOnGuardException);
```

Variables

StrRes : TOgStringResource;

Routines

Generate Key Routines

- GenerateRandomKeyPrim
- GenerateMD5KeyPrim
- GenerateTMDKeyPrim

Modifier Routines

- ApplyModifierToKeyPrim
- CreateMachineID
- GenerateDateModifierPrim
- GenerateMachineModifierPrim
- GenerateStringModifierPrim
- GenerateUniqueModifierPrim

Hash Routines

- StringHashElf

Mixing Routines

- MixBlock

Utility Routines

- ExpandDate
- ShrinkDate
- BufferToHex
- BufferToHexBytes
- HexStringIsZero
- HexToBuffer
- GetCodeType
- GetExpirationDate
- OgFormatDate
- Max
- Min
- XorMem

- MyHashElf
- GetDiskSerialNumber
- GetDriveType

- HiWord
- CoCreateGuid
- timeGetTime

Date Code

- GetDateCodeValue
- InitDateCode
- IsDateCodeExpired
- IsDateCodeValid
- GetDateCodeStart
- GetDateCodeEnd
- InitDateCodeEx

Days Code

- DecDaysCode
- GetDaysCodeValue
- InitDaysCode
- IsDaysCodeExpired
- IsDaysCodeValid

Registration Code

- InitRegCode
- IsRegCodeExpired
- IsRegCodeValid
- IsRegCodeRegisteredTo

Serial Number Code

- GetSerialNumberCodeValue
- InitSerialNumberCode
- IsSerialNumberCodeExpired
- IsSerialNumberCodeValid

Special Code

- GetSpecialCodeValue
- InitSpecialCode
- IsSpecialCodeExpired

- IsSpecialCodeValid

Usage Code

- DecUsageCode
- GetUsageCodeValue
- InitUsageCode
- IsUsageCodeExpired
- IsUsageCodeValid

```
{$IFDEF Win16}
function GetDiskSerialNumber(Drive : AnsiChar) : LongInt;
{$ENDIF}
{$IFDEF LINUX}
function GetDiskSerialNumber(Drive : AnsiChar) : LongInt;
function MyHashElf(const Buf;  BufSize : LongInt) : LongInt;
{$ENDIF}
function Max(A, B : LongInt): LongInt;
function Min(A, B : LongInt) : LongInt;
procedure XorMem(var Mem1; const Mem2; Count : Cardinal);
function OgFormatDate(Value : TDateTime) : string;                    {!!.09}


{$IFDEF KYLIX}
function GetDriveType(drive:Integer): Integer;
function HiWord(I: DWORD):Word;
function CoCreateGuid(out guid: TGUID): HResult;
function timeGetTime: DWord;
{$ENDIF}
{$IFDEF FPC}
{$IFDEF LINUX}
function GetDriveType(drive:Integer): Integer;
function HiWord(I: DWORD):Word;
function CoCreateGuid(out guid: TGUID): HResult;
function timeGetTime: Cardinal;
{$ENDIF}
{$IFDEF FREEBSD}
function GetDriveType(drive:Integer): Integer;
```

```
function HiWord(I: DWORD):Word;
function CoCreateGuid(out guid: TGUID): HResult;
function timeGetTime: Cardinal;
{$ENDIF}
{$ENDIF}
```

The OnGuard unit provides all of the code components except for TOgNetCode.

Classes

TOgCodeBase

Components

TOgMakeKeys
TOgMakeCodes

TOgDateCode
TOgDaysCode
TOgNetCode
TOgRegistrationCode
TOgSerialNumberCode
TOgSpecialCode
TOgUsageCode

The OgFirst unit provides routines that allow you to detect when a second instance of an application is being executed and to force the previous instance of the application to become the active application.

ActivateFirstInstance

IsFirstInstance

The OgNetWrk unit provides the network access component, classes, types and API routines.

Types

TNetAccess
TNetAccessInfo
TGetFileNameEvent

Components

TOgNetCode

Routines

CheckNetAccessFile
CreateNetAccessFile
CreateNetAccessFileEx
DecodeNAFCountCode
EncodeNAFCountCode
GetNetAccessFileInfo
IsAppOnNetwork
LockNetAccessFile
ResetNetAccessFile
UnlockNetAccessFile

This unit contain file related routines formerly located in ogutil.

GetFileSize
LockFile
UnlockFile
FlushFileBuffers

Enter topic text here.

Types

```
PSignatureRec = ^TSignatureRec;
TSignatureRec = packed record
  Sig1   : DWord;                    {!!.07}
  Sig2   : DWord;                    {!!.07}
  Sig3   : DWord;                    {!!.07}
  Offset : DWord;                    {!!.07}
  Size   : DWord;                    {!!.07}
  CRC    : DWord;                    {!!.07}
  Sig4   : DWord;                    {!!.07}
  Sig5   : DWord;                    {!!.07}
  Sig6   : DWord;                    {!!.07}
end;

TExeStatus = (
  exeSuccess,          {no error}
  exeSizeError,        {the file size has changed}
  exeIntegrityError,   {CRC does not match}
  exeNotStamped,       {the exe has not been stamped}
  exeAccessDenied      {share violation}                {!!.05}
);

TCheckedExeEvent = procedure(Sender : TObject; Status : TExeStatus) of
object;
```

Classes

TOgProtectExe

Routines

IsExeTampered
ProtectExe
UnprotectExe
UpdateChecksum
FileCRC32
UpdateCRC32

This unit contains the TKeyGenerateFrm class.

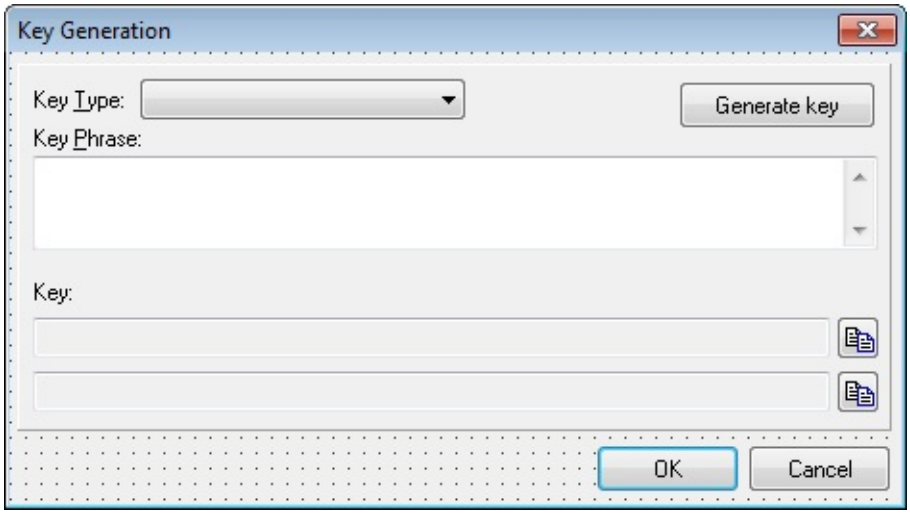
The Key Type combo box contains the options:

- Random
- Standard Text
- Case-sensitive Text

These values correspond to TKeyType.

VCL = OnGuard1.dfm
CLX = QOnGuard1.xfm
Lazarus = lcl\QOnGuard1

VCL


Key Generation
Key Type:
Generate key
Key Phrase:
Key:
OK
Cancel

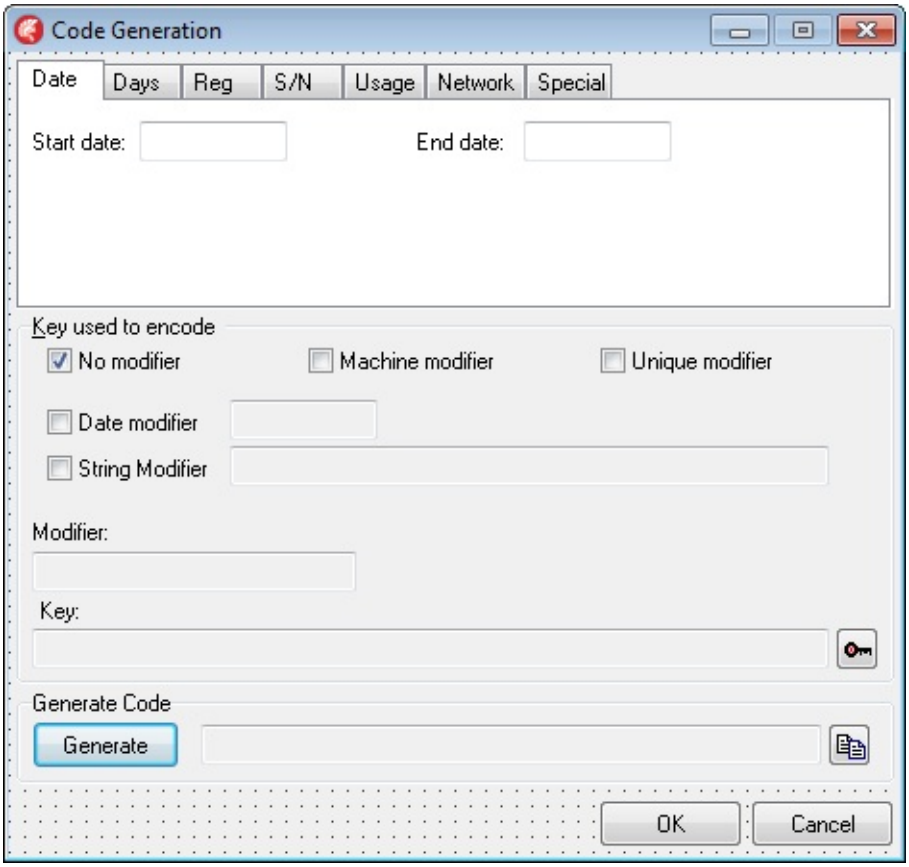
This unit contains the TCodeGenerateFrm class.

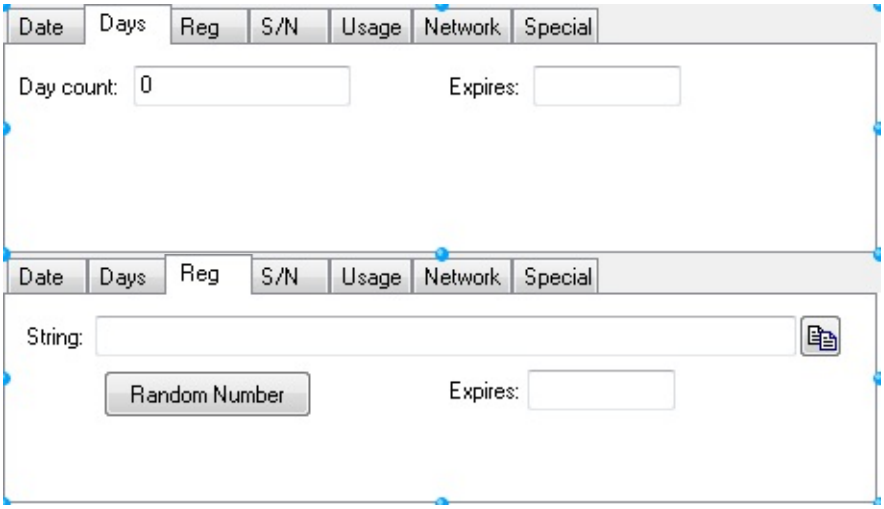
The tabs across the top represent the code type and must match the sequence in TCodeType.

Clicking on the button will open the key maintenance form in OnGuard3.

VCL = OnGuard2.dfm
CLX = QOnGuard2.xfm
Lazarus = lcl\QOnGuard2

VCL


Code Generation
Date
Days
Reg
S/N
Usage
Network
Special
Start date:
End date:
Key used to encode
No modifier
Machine modifier
Unique modifier
Date modifier
String Modifier
Modifier:
Key:
Generate Code
Generate
OK
Cancel


Date
Days
Reg
S/N
Usage
Network
Special
Day count:
0
Expires:
Date
Days
Reg
S/N
Usage
Network
Special
String:
Random Number
Expires:

Date | Days | Reg | S/N | Usage | Network | Special

Serial Number: 0 Expires:

Random Number

Date | Days | Reg | S/N | Usage | Network | Special

Usage count: 0 Expires:

Date | Days | Reg | S/N | Usage | Network | Special

Access Slots: 2

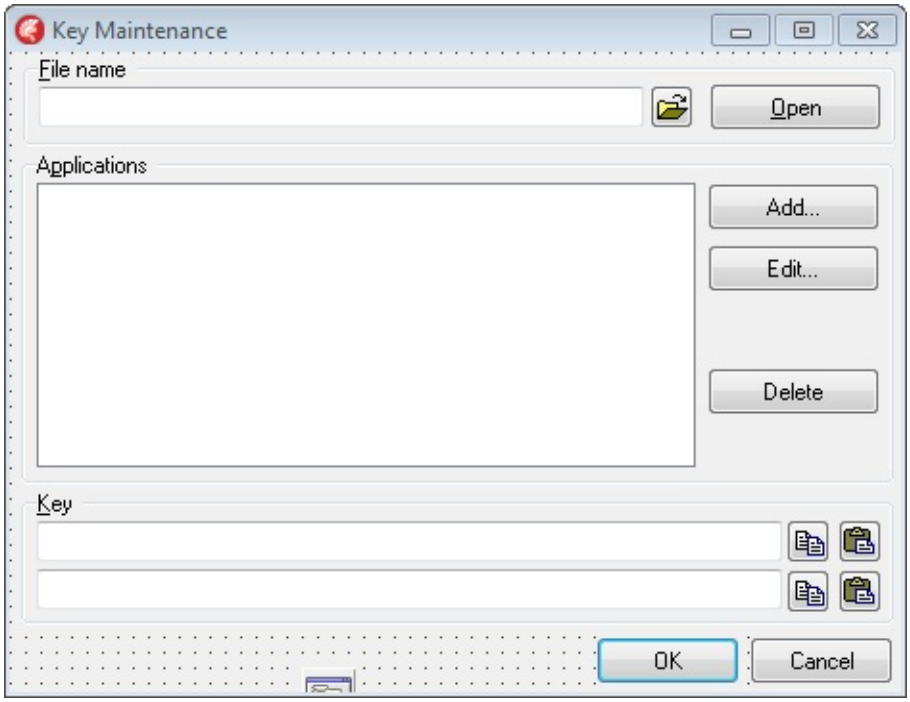
Date | Days | Reg | S/N | Usage | Network | Special

Special data: 0 Expires:

This unit contains the TKeyMaintFrm class.

VCL = OnGuard3.dfm
CLX = QOnGuard3.xfm
Lazarus = lcl\QOnGuard3

VCL

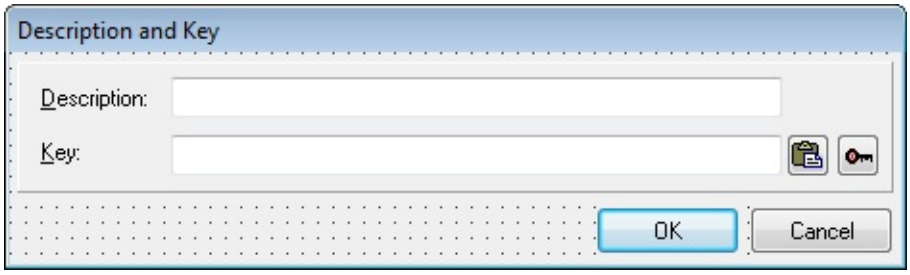

Key Maintenance
File name
Open
Applications
Add...
Edit...
Delete
Key
OK
Cancel

This unit contains the TEditProductFrm class.

Clicking on the  button will open the key generation form in OnGuard1.

VCL = OnGuard4.dfm
CLX = QOnGuard4.xfm
Lazarus = lcl\QOnGuard4

VCL



The OnGuard5 unit contains the class TOgCodeProperty which is used as a Property Editor in the IDE.

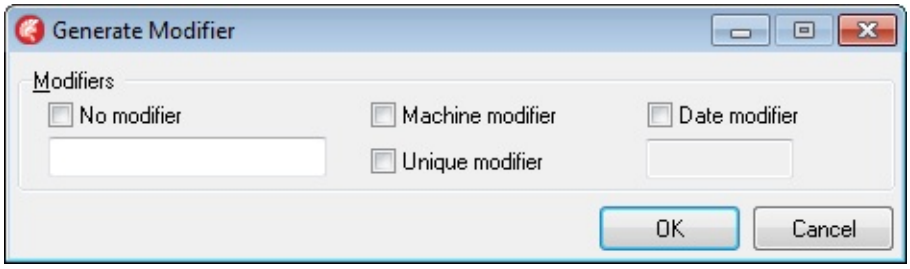
The TOgCodeProperty.Edit method uses the TCodeGenerateFrm class found in OnGuard2.

CLX = QOnGuard5
Lazarus = lcl\QOnGuard5

The OnGuard6 unit contains the TModifierFrm class and the TOgModifierProperty class which is used as a Property Editor in the IDE.

VCL = OnGuard6.dfm
CLX = QOnGuard6.xfm
Lazarus = lcl\QOnGuard6

VCL


Generate Modifier
Modifiers
No modifier
Machine modifier
Date modifier
Unique modifier
OK
Cancel

The OnGuard7 unit contains the class TOgFileNameProperty which is used as a Property Editor in the IDE.

CLX = QOnGuard7
Lazarus = lcl\QOnGuard7

This unit provides the About dialog.
It also provides the TOgAboutProperty which is used as a Property Editor in the IDE.

VCL = OgAbout0.dfm
CLX = QOgAbout0.xfm
Lazarus = lcl\QOgAbout0

VCL


About TurboPower OnGuard
OnGuard
Version 1.00
OnGuard home page:
http://sourceforge.net/projects/tponguard
Released under the Mozilla License 1.1
(MPL 1.1)
Online newsgroups
https://sourceforge.net/forum/?group_id=71010
Copyright (C) 1997-2002, TurboPower Software Company.
All rights reserved.
OK

The OgReg unit contains the TOgCodeGenEditor class which is used as a Property Editor in the IDE.
This unit also exposes the Register procedure used to register the components in Delphi.

The register procedure adds a component editor to TOgCodeBase with two actions: Generate Code and Generate Key.

TOgCodeBase is also given property editors:

- Code = TOgCodeProperty
- Modifier = TOgModifierProperty
- About = TOgAboutProperty

TOgProtectExe, TOgMakeCodes, and TOgMakeKeys are given the TOgAboutProperty property editor.

TOgMakeCodes and TOgMakeKeys are given the TOgFileNameProperty property editor on the KeyFileName property.

TOgCodeBase Properties

- AutoCheck
- Code
- Modifier
- StoreCode
- StoreModifier

TOgCodeBase Events

- OnChecked
- OnGetKey
- OnGetCode
- OnGetModifier

TOgCodeBase Methods

- CheckCode
- IsCodeValid

TOgDateCode Properties

AutoCheck
Code
Modifier
StoreCode
StoreModifier

TOgDateCode Events

OnChecked
OnGetKey
OnGetCode
OnGetModifier

TOgDateCode Methods

CheckCode

GetValue

IsCodeValid

TOgDaysCode Properties

AutoCheck
AutoDecrease
Code
StoreCode

TOgDaysCode Events

- OnChangeCode
- OnChecked
- OnGetKey
- OnGetCode
- OnGetModifier

TOgDaysCode Methods

- CheckCode
- Decrease
- GetValue
- IsCodeValid

TOgMakeCodes Properties

- Code
- CodeType
- Key
- KeyFileName
- KeyType
- ShowHints

TOgMakeCodes Methods

- Execute

TOgMakeKeys Properties

- Key
- KeyFileName
- KeyType
- ShowHints

TOgMakeKeys Methods

- ApplyModifierToKey
- Execute
- GenerateDateModifier
- GenerateKey
- GenerateMachineModifier
- GenerateRandomKey
- GenerateStringModifier
- GenerateUniqueModifier

TOgNetCode Properties

- ActiveUsers
- AutoCheck
- Code
- FileName
- InvalidUsers
- MaxUsers
- Modifier
- StoreCode
- StoreModifier

TOgNetCode Events

OnChecked
OnGetKey
OnGetCode
OnGetModifier

TOgNetCode Methods

CheckCode
IsCodeValid
IsRemoteDrive
ResetAccessFile

TOgProtectExe Properties

- AutoCheck
- CheckSize

TOgProtectExe Events

- OnChecked

TOgProtectExe Methods

- CheckExe
- StampExe
- UnStampExe

TOgRegistrationCode Properties

AutoCheck
Code
Modifier
RegString
StoreCode
StoreModifier
StoreRegString

TOgRegistrationCode Events

OnChecked
OnGetKey
OnGetCode
OnGetModifier
OnGetRegString

TOgRegistrationCode Methods

CheckCode
IsCodeValid

TOgSerialNumberCode Properties

AutoCheck
Code
Modifier
StoreCode
StoreModifier

TOgSerialNumberCode Events

OnChecked
OnGetKey
OnGetCode
OnGetModifier

TOgSerialNumberCode Methods

CheckCode

GetValue

IsCodeValid

TOgSpecialCode Properties

AutoCheck
Code
Modifier
StoreCode
StoreModifier

TOgSpecialCode Events

OnChecked
OnGetKey
OnGetCode
OnGetModifier

TOgSpecialCode Methods

CheckCode

GetValue

IsCodeValid

TOgUsageCode Properties

AutoCheck
AutoDecrease
Modifier
StoreModifier

TOgUsageCode Events

- OnChangeCode
- OnChecked
- OnGetKey
- OnGetCode
- OnGetModifier

TOgUsageCode Methods

- CheckCode
- Decrease
- GetValue
- IsCodeValid

GenerateRandomKeyPrim

procedure GenerateRandomKeyPrim (**var** Key; KeySize : Cardinal);

GenerateRandomKeyPrim produces a Key using a random numbers.

This routine is defined in the OnGuard unit. (as of 1.15 this routine has been moved to the OgUtil unit)

GenerateMD5KeyPrim

procedure GenerateMD5KeyPrim (**var** Key: TKey; **const** Str : **string**);

GenerateMD5KeyPrim produces a Key by applying the MD5 hash to the string passed as Str

The routine is case sensitive.

This routine is defined in the OnGuard unit. (as of 1.15 this routine has been moved to the OgUtil unit)

GenerateTMDKeyPrim

procedure GenerateTMDKeyPrim (**var** Key; KeySize : Cardinal; **const** Str : **string**);

GenerateTMDKeyPrim produces key by applying a hash algorithm to the string passed in Str.

This routine is defined in the OnGuard unit. (as of 1.15 this routine has been moved to the OgUtil unit)

ApplyModifierToKeyPrim

procedure ApplyModifierToKeyPrim (Modifier : LongInt;
var Key; KeySize : Cardinal);

ApplyModifierToKeyPrim Xor's the Modifier value with the Key returning the modified key as the Key parameter.

Use this routine to *sign* a key.

KeySize if the size of the key in bytes

This routine is defined in the OnGuard unit. (as of 1.15 this routine has been moved to the OgUtil unit)

CreateMachineID

This is a private function first added in version 1.05.
In version 1.14 the Ansi parameter was added to the Win32 version.

function CreateMachineID(MachineInfo : TEsMachineInfoSet; Ansi: Boolean = True) : LongInt;
function CreateMachineID(MachineInfo : TEsMachineInfoSet) : LongInt;

Originally declared in OnGuard it was moved to OgUtil in version 1.15.

Summary

	Delphi							FPC	
	Win16	**Win32**	**Win64**	**MacOS**	**iOS**	**Android**	**Linux**	**UNIX**	**Win32**
midUser	n/a	Yes	n/a	n/a	n/a	n/a	n/a	Yes	Yes
midSystem	Yes	Yes	n/a	n/a	n/a	n/a	n/a	Yes	Yes
midNetwork	Yes	Yes	n/a	n/a	n/a	n/a	n/a	Yes	Yes
midDrives	Yes	Yes	n/a	n/a	n/a	n/a	n/a	n/a	Yes
Following added in version 1.15									
midCPUID	n/a	Yes	n/a	n/a	n/a	n/a	n/a	n/a	?
midCPUIDJCL	n/a	?	n/a	n/a	n/a	n/a	n/a	n/a	?
midBIOS	n/a	Yes	n/a	n/a	n/a	n/a	n/a	n/a	?
midWinProd	n/a	Yes	n/a	n/a	n/a	n/a	n/a	n/a	?
midCryptoID	n/a	Yes	n/a	n/a	n/a	n/a	n/a	n/a	?
midNetMAC	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
midDomain	n/a	Yes	n/a	n/a	n/a	n/a	n/a	n/a	?

MachineInfo	**Comments**
midUser	on Win32 uses the HKEY_LOCAL_MACHINE registry hive to read the values in Software\Microsoft\Windows\CurrentVersion or Software\Microsoft\Windows NT\CurrentVersion. Uses the values of RegisteredOwner and RegisteredOrganization. on FPC uses Environment variables USERNAME, USER, or LOGNAME.
midSystem	on Win16 uses the Windows directory, Windows System directory, GetWinFlags, and WinProcs.GetVersion. on Win32 uses GetSystemInfo's dwOemId and dwProcessorType values. on FPC uses /proc/cpuinfo, /proc/sys/kernel/version, /proc/sys/kernel/osrelease, /proc/sys/kernel/hostname files. on Kylix uses /proc/cpuinfo, /proc/sys/kernel/version, /proc/sys/kernel/osrelease, /proc/sys/kernel/hostname files.

midNetwork	on Win16 compares the Data4 field of two GUIDs. If the same then uses the Data4 field. on Win32 compares the Data4 field of two GUIDs. If the same then uses the Data4 field. on FPC compares the Data4 field of two GUIDs. If the same then uses the Data4 field. on Kylix compares the Data4 field of two GUIDs. If the same then uses the Data4 field.
midDrives	on Win16 uses the GetDiskSerialNumber for each fixed drive from C: to Z:. on Delphi-Win32 uses the GetVolumeInformation for each fixed drive from C: to Z:. Ignores SUBST drives. on Kylix uses the GetDiskSerialNumber for each fixed drive from 2 to 26 (C to Z).
midCPUID	on Win32 uses the HKEY_LOCAL_MACHINE registry hive to read the values in Software\Microsoft\Windows NT\CurrentVersion. Uses the values of Identifier, ProcessorNameString, and VendorIdentifier.
midCPUIDJCL	
midBIOS	on Win32 uses the HKEY_LOCAL_MACHINE registry hive to read the values in HARDWARE\DESCRIPTION\System\BIOS. Uses the values of BaseBoardManufacturer, BaseBoardProduct, BaseBoardVersion, BIOSReleaseDate, BIOSVendor, BIOSVersion, SystemFamily, SystemManufacturer, SystemProductName, SystemSKU, and SystemVersion.
midWinProd	on Win32 uses the HKEY_LOCAL_MACHINE registry hive to read the values in Software\Microsoft\Windows NT\CurrentVersion. Uses the values of ProductID, InstallDate, ProductName, InstallationType and EditionID.
midCryptoID	on Win32 uses the HKEY_LOCAL_MACHINE registry hive to read the values in Software\Microsoft\Cryptography. Uses the value of MachineGUID.
midNetMAC	
midDomain	on Win32 uses the HKEY_LOCAL_MACHINE registry hive to read the values in

	System\CurrentControlSet\Services\Tcpip\Parameters. Uses the values of DhcpDomain, Domain, ICSDomain, and "NV Domain".

GenerateDateModifierPrim

function GenerateDateModifierPrim (D : TDateTime) : LongInt;

GenerateDateModifierPrim produces a key modifier based on the date D.

This routine is defined in the OnGuard unit. (as of 1.15 this routine has been moved to the OgUtil unit)

GenerateMachineModifierPrim

function GenerateMachineModifierPrim: LongInt;

GenerateMachineModifierPrim produces a key modifier based on specific hardware information.

Information about hard disk capacity, network card serial number, and other items specific to a particular computer are used to create this value.

This function calls CreateMachineID using midSystem, midUser, and midDrives.

This routine is defined in the OnGuard unit. (as of 1.15 this routine has been moved to the OgUtil unit)

GenerateStringModifierPrim

function GenerateStringModifierPrim (**const** S : **string**) : LongInt;

GenerateStringModifierPrim produces a key modifier by applying a hash algorithm to the string passed in S.

This routine is case sensitive.

This routine is defined in the OnGuard unit. (as of 1.15 this routine has been moved to the OgUtil unit)

GenerateUniqueModifierPrim

function GenerateUniqueModifierPrim: LongInt;

GenerateUniqueModifierPrim produces a key modifier using random numbers.

This routine is defined in the OnGuard unit. (as of 1.15 this routine has been moved to the OgUtil unit)

StringHashElf

function StringHashElf (**const** Str : string) : LongInt;

StringHashElf produces a hash value based on the text passed in Str.

This routine is defined in the OnGuard unit. (as of 1.15 this routine has been moved to the OgUtil unit)

MixBlock

Enter topic text here.

ExpandDate

function ExpandDate (D : Word) : TDateTime;

ExpandDate translates an OnGuard date offset to an actual date.

OnGuard uses a date offset to reduce the amount of space necessary to store a date. OnGuard creates a date offset by subtracting the TDateTime value for 1 January 1996 from the actual date.

Exceptions to the conversion rules are that a value of 0 expands to 1 January 9999 and date offsets larger than 65535 are represented as 0 (anything after 6 June 2175).

This routine is defined in the OnGuard unit. (as of 1.15 this routine has been moved to the OgUtil unit)

ShrinkDate

function ShrinkDate (D : TDateTime) : Word;

ShrinkDate translates a date to an OnGuard date offset.

OnGuard uses a date offset to reduce the amount of space necessary to store a date. OnGuard creates a date offset by subtracting the TDateTime value for 1 January 1996 from the actual date.

Exceptions to the conversion rules are that a value of 0 expands to 1 January 9999 and date offsets larger than 65535 are represented as 0 (anything after 6 Jun 2175).

This routine is defined in the OnGuard unit. (as of 1.15 this routine has been moved to the OgUtil unit)

BufferToHex

function BufferToHex (**const** Buf; BufSize : Cardinal) : **string**;

BufferToHex converts one or more bytes to hex.

Buf contains one or more bytes and BufSize if the number of bytes in Buf. The hexadecimal version of Buf is returned as the function result.

This routine is defined in the OgUtil unit.

BufferToHexBytes

function BufferToHexBytes (**const** Buf; BufSize : Cardinal) : **string**;

BufferToHexBytes performs the same operation as the BufferToHex function except that the function result is formatted to represent an array of hexadecimal bytes separated by commas.

Example result: "$02, $67, $FF"

This routine is defined in the OgUtil unit.

HexStringIsZero

function HexStringIsZero (**const** Hex : **string**) : Boolean;

HexStringIsZero returns *true* if the hexadecimal string passed as Hex consists entirely of 0's, otherwise *false*.

This routine is defined in the OgUtil unit.

HexToBuffer

function HexToBuffer (**const** Hex : **string**; **var** Buf; BufSize : Cardinal) : Boolean;

HexToBuffer converts the hexadecimal string in Hex to bytes that are stored in Buf.

Punctuation ($, spaces, commas, parentheses, ...) is ignored.

BufSize is the number of bytes to store in Buf and must be the number of hexadecimal bytes in Hex. If an error occurs, *false* is returned, otherwise *true*.

This routine is defined in the OgUtil unit.

GetCodeType

function GetCodeType (**const** Key : TKey; **const** Code : TCode) : TCodeType;

TCodeType = (ctDate, ctDays, ctRegistration, ctSerialNumber, ctUsage, ctNetwork, ctSpecial, ctUnknown);

GetCodeType returns the type of code passed as the Code parameter.

Key must be the same key that was used when the code was created or ctUnknown is returned.

This routine is defined in the OnGuard unit. (as of 1.15 this routine has been moved to the OgUtil unit)

GetExpirationDate

function GetExpirationDate (**const** Key : TKey; **const** Code : TCode) : TDateTime;

GetExpirationDate returns the date that the code passed as the Code parameter expires.

If the code has no expiration date or is invalid, 1 January 9999 is returned.Key must be the same key that was used to create the code or the code is considered invalid.

This routine is defined in the OnGuard unit. (as of 1.15 this routine has been moved to the OgUtil unit)

As of version 1.15 this function checks the expiration field for date codes. If the expiration field is not zero then return it otherwise return the EndDate field like previous versions did.

GetDateCodeValue

function GetDateCodeValue (**const** Key : TKey; **const** Code : TCode) : TDateTime;

GetDateCodeValue returns the expiration date stored in the Code.

Key must be the same key that was used to create the code or the code is considered invalid. If the code is invalid, 1 January 9999 is returned.

This routine is defined in the OnGuard unit. (as of 1.15 this routine has been moved to the OgUtil unit)

InitDateCode

procedure InitDateCode (**const** Key : TKey; StartDate, EndDate : TDateTime; **var** Code : TCode);

InitDateCode creates and initializes a date code using Key, StartDate, and EndDate.

The resulting code is valid for dates between StartDate and EndDate inclusive.

This routine is defined in the OnGuard unit. (as of 1.15 this routine has been moved to the OgUtil unit)

With version 1.15 the StartDate is checked.

An exception is generated if the StartDate is less than or equal to the BaseDate or if it is greater than 2175-Jun-6.

IsDateCodeExpired

function IsDateCodeExpired (**const** Key : TKey; **const** Code : TCode) : Boolean;

IsDateCodeExpired returns *true* if the Code has expired, otherwise *false*.

Key must be the same key that was used to create the code or the code is considered invalid. If the code is invalid, this function returns *true*.

This routine is defined in the OnGuard unit. (as of 1.15 this routine has been moved to the OgUtil unit)

IsDateCodeValid

function IsDateCodeValid (**const** Key : TKey; **const** Code : TCode) : Boolean;

IsDateCodeValid returns *true* if Code is a valid date code, otherwise *false*.

Key must be the same key that was used to create the code or the code is considered invalid.

This routine is defined in the OnGuard unit. (as of 1.15 this routine has been moved to the OgUtil unit)

InitDateCodeEx

procedure InitDateCodeEx (**const** Key : TKey; StartDate, EndDate, Expires : TDateTime; **var** Code : TCode);

InitDateCodeEx creates and initializes a date code using Key, StartDate, EndDate, and Expires.

The resulting code is valid for dates between StartDate and EndDate inclusive.

The difference between this function and InitDateCode is the addition of an expiration date.

This routine is defined in the OgUtil unit.

Added in version 1.15.

An exception is generated if the StartDate is less than or equal to the BaseDate or if it is greater than 2175-Jun-6.

An exception is generated if Expires is less than or equal to the BaseDate or if it is less than or equal to the StartDate.

GetDateCodeStart

function GetDateCodeStart (**const** Key : TKey; **const** Code : TCode) : TDateTime;

GetDateCodeStart returns the start date stored in the Code.

Key must be the same key that was used to create the code or the code is considered invalid. If the code is invalid, 1 January 9999 is returned.

This routine is defined in the OgUtil unit.

Added in version 1.15.

GetDateCodeEnd

function GetDateCodeEnd (**const** Key : TKey; **const** Code : TCode) : TDateTime;

GetDateCodeEnd returns the end date stored in the Code.

Key must be the same key that was used to create the code or the code is considered invalid. If the code is invalid, 1 January 9999 is returned.

This routine is defined in the OgUtil unit.

Added in version 1.15.

DecDaysCode

procedure DecDaysCode (**const** Key : TKey; **var** Code : TCode);

DecDaysCode reduces the internal "days count" value by one and returns the modified code as the Code parameter.

Key must be the same key that was used to create the code or the code is considered invalid.

This routine is defined in the OnGuard unit. (as of 1.15 this routine has been moved to the OgUtil unit)

GetDaysCodeValue

function GetDaysCodeValue (**const** Key : TKey; **const** Code : TCode) : LongInt;

GetDaysCodeValue returns the expiration date stored in the Code.

Key must be the same key that was used to create the code or the code is considered invalid. If the code is invalid, 0 is returned.

This routine is defined in the OnGuard unit. (as of 1.15 this routine has been moved to the OgUtil unit)

InitDaysCode

procedure InitDaysCode (**const** Key : TKey; Days : Word; Expires : TDateTime; **var** Code : TCode);

InitDaysCode creates and initializes a days code using Key, Days, and Expires.

Days is stored as part of the Code.

The resulting code is valid for the number of days of use specified in the Days parameter and until the date stored in Expires is reached.

This routine is defined in the OnGuard unit. (as of 1.15 this routine has been moved to the OgUtil unit)

IsDaysCodeExpired

function IsDaysCodeExpired (**const** Key : TKey; **const** Code : TCode) : Boolean;

IsDaysCodeExpired returns *true* if the Code has expired, otherwise *false*.

Key must be the same key that was used to create the code or the code is considered invalid. If the code is invalid, this function returns *true*.

This routine is defined in the OnGuard unit. (as of 1.15 this routine has been moved to the OgUtil unit)

IsDaysCodeValid

function IsDaysCodeValid (**const** Key : TKey; **const** Code : TCode) : Boolean;

IsDaysCodeValid returns *true* if Code is a valid days code, otherwise *false*.

Key must be the same key that was used to create the code or the code is considered invalid.

This routine is defined in the OnGuard unit. (as of 1.15 this routine has been moved to the OgUtil unit)

InitRegCode

procedure InitRegCode (**const** Key : TKey; **const** RegStr : **string**; Expires : TDateTime; **var** Code : TCode);

InitRegCode creates and initializes a registration code using Key, RegStr, and Expires.

The code stores a hash value that was derived from RegStr. RegStr cannot be extracted from the code.

The resulting code is valid until the date stored in Expires is reached.

This routine is defined in the OnGuard unit. (as of 1.15 this routine has been moved to the OgUtil unit)

IsRegCodeExpired

function IsRegCodeExpired (**const** Key : TKey; **const** Code : TCode) : Boolean;

IsRegCodeExpired returns *true* if the Code has expired, otherwise *false*.

Key must be the same key that was used to create the code or the code is considered invalid. If the code is invalid, this function returns *true*.

This routine is defined in the OnGuard unit. (as of 1.15 this routine has been moved to the OgUtil unit)

IsRegCodeValid

function IsRegCodeValid (**const** Key : TKey; **const** Code : TCode) : Boolean;

IsRegCodeValid returns *true* if Code is a valid registration code, otherwise *false*.

Key must be the same key that was used to create the code or the code is considered invalid.

This routine is defined in the OnGuard unit. (as of 1.15 this routine has been moved to the OgUtil unit)

IsRegCodeRegisteredTo

function IsRegCodeRegisteredTo(**const** Key : TKey; **const** Code : TCode; **const** RegStr: AnsiString) : Boolean;

IsRegCodeRegisteredTo returns *true* if Code is a valid registration code and the registration sting matches, otherwise *false*.

Key must be the same key that was used to create the code or the code is considered invalid.

This routine is defined in the OgUtil unit.

Added in version 1.15.

GetSerialNumberCodeValue

function GetSerialNumberCodeValue (**const** Key : TKey; **const** Code : TCode) : LongInt;

GetSerialNumberCodeValue returns the serial number that was used to create the Code.

Key must be the same key that was used to create the code or the code is considered invalid. If the code is invalid, 0 is returned.

This routine is defined in the OnGuard unit. (as of 1.15 this routine has been moved to the OgUtil unit)

InitSerialNumberCode

procedure InitSerialNumberCode (**const** Key : TKey; Serial : LongInt; Expires : TDateTime; **var** Code : TCode);

InitSerialNumberCode creates and initializes a serial number code using Key, Serial, and Expires.

Serial is stored as part of the Code.

The resulting code is valid until the date stored in Expires is reached.

This routine is defined in the OnGuard unit. (as of 1.15 this routine has been moved to the OgUtil unit)

IsSerialNumberCodeExpired

function IsSerialNumberCodeExpired (**const** Key : TKey; **const** Code : TCode) : Boolean;

IsSerialNumberCodeExpired returns *true* if the Code has expired, otherwise *false*.

Key must be the same key that was used to create the code or the code is considered invalid. If the code is invalid, this function returns *true*.

This routine is defined in the OnGuard unit. (as of 1.15 this routine has been moved to the OgUtil unit)

IsSerialNumberCodeValid

function IsSerialNumberCodeValid (**const** Key : TKey; **const** Code : TCode) : Boolean;

IsSerialNumberCodeValid returns *true* if Code is a valid serial number code, otherwise *false*.

Key must be the same key that was used to create the code or the code is considered invalid.

This routine is defined in the OnGuard unit. (as of 1.15 this routine has been moved to the OgUtil unit)

IsSerialNumberCodeValidFor

function IsSerialNumberCodeValid (**const** Key : TKey; **const** Code : TCode; **const** SerialNumber : LongInt) : Boolean;

IsSerialNumberCodeValidFor returns *true* if Code is a valid serial number code and the SerialNumber matches, otherwise *false*.

Key must be the same key that was used to create the code or the code is considered invalid.

This routine is defined in the OgUtil unit.

Added in version 1.15.

GetSpecialCodeValue

function GetSpecialCodeValue (**const** Key : TKey; **const** Code : TCode) : LongInt;

GetSpecialCodeValue returns the value that was used to create the Code.

Key must be the same key that was used to create the code or the code is considered invalid. If the code is invalid, 0 is returned.

This routine is defined in the OnGuard unit. (as of 1.15 this routine has been moved to the OgUtil unit)

InitSpecialCode

procedure InitSpecialCode (**const** Key : TKey; Value : LongInt; Expires : TDateTime; **var** Code : TCode);

InitSpecialCode creates and initializes a special code using Key, Value, and Expires.

Value is stored as part of the Code.

The resulting code is valid until the date stored in Expires is reached.

This routine is defined in the OnGuard unit. (as of 1.15 this routine has been moved to the OgUtil unit)

IsSpecialCodeExpired

function IsSpecialCodeExpired (**const** Key : TKey; **const** Code : TCode) : Boolean;

IsSpecialCodeExpired returns *true* if the Code has expired, otherwise *false*.

Key must be the same key that was used to create the code or the code is considered invalid. If the code is invalid, this function returns *true*.

This routine is defined in the OnGuard unit. (as of 1.15 this routine has been moved to the OgUtil unit)

IsSpecialCodeValid

function IsSpecialCodeValid (**const** Key : TKey; **const** Code : TCode) : Boolean;

IsSpecialCodeValid returns *true* if Code is a valid special code, otherwise *false*.

Key must be the same key that was used to create the code or the code is considered invalid.

This routine is defined in the OnGuard unit. (as of 1.15 this routine has been moved to the OgUtil unit)

IsSpecialCodeValidFor

function IsSpecialCodeValid (**const** Key : TKey; **const** Code : TCode; **const** Value: LongInt) : Boolean;

IsSpecialCodeValidFor returns *true* if Code is a valid special code and the Value matches, otherwise *false*.

Key must be the same key that was used to create the code or the code is considered invalid.

This routine is defined in the OgUtil unit.

Added in version 1.15.

DecUsageCode

procedure DecUsageCode (**const** Key : TKey; **var** Code : TCode);

DecUsageCode reduces the internal "usage count" value by one and returns the modified code as the Code parameter.

Key must be the same key that was used to create the code or the code is considered invalid.

This routine is defined in the OnGuard unit. (as of 1.15 this routine has been moved to the OgUtil unit)

In version 1.15:

If the conditional define OgUsageUnlimited is enabled then a check is made to see if the usage count = 65535 and expiration = 65535 and last change = 1 is set.
If all three conditions are true then the code is treated as an unlimited usage code in which case it is not decremented nor is the last updated date changed.

GetUsageCodeValue

function GetUsageCodeValue (**const** Key : TKey; **const** Code : TCode) : LongInt;

GetUsageCodeValue returns the current usage count value store in the Code.

Key must be the same key that was used to create the code or the code is considered invalid. If the code is invalid, 0 is returned.

This routine is defined in the OnGuard unit. (as of 1.15 this routine has been moved to the OgUtil unit)

InitUsageCode

procedure InitUsageCode (**const** Key : TKey; Count : LongInt; Expires : TDateTime; **var** Code : TCode);

InitUsageCode creates and initializes a usage code using Key, Count, and Expires.

Count is stored as part of the Code.

The resulting code is valid until the internal Count is 0 or the date stored in Expires is reached.

This routine is defined in the OnGuard unit. (as of 1.15 this routine has been moved to the OgUtil unit)

IsUsageCodeExpired

function IsUsageCodeExpired (**const** Key : TKey; **const** Code: TCode) : Boolean;

IsUsageCodeExpired returns *true* if the Code has expired, otherwise *false*.

Key must be the same key that was used to create the code or the code is considered invalid. If the code is invalid, this function returns *true*.

This routine is defined in the OnGuard unit. (as of 1.15 this routine has been moved to the OgUtil unit)

IsUsageCodeValid

function IsUsageCodeValid (**const** Key : TKey; **const** Code : TCode) : Boolean;

IsUsageCodeValid returns *true* if Code is a valid usage code, otherwise *false*.

Key must be the same key that was used to create the code or the code is considered invalid.

This routine is defined in the OnGuard unit. (as of 1.15 this routine has been moved to the OgUtil unit)

InitUsageCodeUnlimited

procedure InitUsageCode (**const** Key : TKey; **var** Code : TCode);

InitUsageCodeUnlimited creates and initializes a usage code using Key, Count=65535, Expires=65535, and LastChange=1.

Count is stored as part of the Code.

The resulting code is valid until the internal Count is 0 or the date stored in Expires is reached.

This routine is defined in the OgUtil unit.

Added in 1.15.

Only available if the conditional define OgUsageUnlimited is enabled.

IsAppOnNetwork

function IsAppOnNetwork (**const** ExePath : **string**) : Boolean;

IsAppOnNetwork returns *true* if the drive specified in ExePath is a remote drive, otherwise *false*.

This routine is defined in the OgNetWrk unit.

CheckNetAccessFile

```
function CheckNetAccessFile (const NetAccess :
TNetAccess) : Boolean;
```

```
TNetAccess = packed record
 Fh        : Integer;
 Key       : TKey;
 CheckValue : Word;
 Index     : Word;
end;
```

CheckNetAccessFile verifies that the net access file referenced by NetAccess has at least one slot that is not in use.

If there is at least one open slot in the net access file, CheckNetAccessFile returns *true*, otherwise *false*.

This routine is defined in the OgNetWrk unit.

CreateNetAccessFile

function CreateNetAccessFile **(const** FileName : **string**;
const Key : TKey; Count : Word) : Boolean;

CreateNetAccessFile creates a net access for Count users file using FileName as the name of the file and Key to encode the file.

If a file with FileName as its name exists it is overwritten without warning.

This routine is defined in the OgNetWrk unit.

CreateNetAccessFileEx

function CreateNetAccessFileEx **(const** FileName : **string**; **const** Key : TKey; **const** Code : TCode) : Boolean;

CreateNetAccessFileEx creates a net access file using the access count value from a previously encoded net access Code.

Key must be the same key that was used to create the code or the code is considered invalid.

This routine is defined in the OgNetWrk unit.

DecodeNAFCountCode

function DecodeNAFCountCode (**const** Key : TKey; **const** Code : TCode) : LongInt;

DecodeNAFCountCode uses Key to decode Code and returns the number of authorized users as the function result.

Key must be the same key that was used to create the code or the code is considered invalid. If the code is invalid, 0 is returned.

This routine is defined in the OgNetWrk unit.

GetNetAccessFileInfo

```
function GetNetAccessFileInfo (const FileName : string;
const Key : TKey; var NetAccessInfo : TNetAccessInfo) :
Boolean;

TNetAccessInfo = packed record
 Total   : Cardinal;
 Locked  : Cardinal;
 Invalid : Cardinal;
end;
```

GetNetAccessFileInfo obtains information about the specified network access file.

FileName is the name of an existing network access file and Key is the key that was used to create it. The network access file information is returne as the NetAccessInfo parameter and consists of the total number of access slots, the number of locked slots, and the number of invalid access slots. (An access slot becomes invalid when the application using it is terminated in a non-standard way.)

GetNetAccessFileInfo returns *false* if there was an error, otherwise *true*.

This routine is defined in the OgNetWrk unit.

EncodeNAFCountCode

procedure EncodeNAFCountCode (**const** Key : TKey; Count : Cardinal; **var** Code : TCode);

EncodeNAFCountCode uses Key to create and encode the usage Count value creating a network code.

The resulting code is returned as the Code parameter.

This routine is defined in the OgNetWrk unit.

LockNetAccessFile

function LockNetAccessFile (**const** FileName : string; **const** Key : TKey; **var** NetAccess : TNetAccess) : Boolean;

```
TNetAccess = packed record
 Fh        : Integer;
 Key       : TKey;
 CheckValue : Word;
 Index     : Word;
end;
```

LockNetAccessFile locks an access slot in the network access file specified by FileName and returns *false* if an error occurs.

This routine is defined in the OgNetWrk unit.

ResetNetAccessFile

function ResetNetAccessFile (**const** FileName : **string**; **const** Key : TKey) : Boolean;

ResetNetAccessFile resets invalid access slots by clearing there "in-uses" status.

Access slots that are currently "in-use" are skipped.

This routine is defined in the OgNetWrk unit.

UnlockNetAccessFile

```
function UnlockNetAccessFile (var NetAccess : TNetAccess)
: Boolean;

TNetAccess = packed record
 Fh        : Integer;
 Key       : TKey;
 CheckValue : Word;
 Index     : Word;
end;
```

UnlockNetAccessFile unlocks an access slot in the network access file specified by FileName and returns *false* if an error occurs.

This routine is defined in the OgNetWrk unit.

IsFirstInstance

function OgFirst.IsFirstInstance : Boolean;

IsFirstInstance determines whether this is the first instance of a program.

This method should be called prior to creating any forms so that the application can be terminated if necessary. IsFirstInstance returns True if this is the first instance of the application.

If IsFirstInstance returns False, you can call ActivateFirstInstance to activate the prior instance of the application.

ActivateFirstInstance

procedure OgFirst.ActivateFirstInstance; {32-bit version}

procedure OgFirst.ActivateFirstInstance(const MainWindowCaption, MainWindowClass : string); {16-bit version}

ActivateFirstInstance locates an applications main window and then makes it the active window.

ActivateFirstInstance forces the window with the specified caption and class to the top of the z-Order and gives it the focus. This method is normally called after detecting that a second instance of the application was executed and subsequently halted. Calling ActivateFirstInstance gives the appearance that running the application a second time succeeded.

The 32-bit version of ActivateFirstInstance does not take any parameters and automatically locates the applications main window. The 16-bit version of this routine requires that the class name and caption of the main form be passed as arguments.

TLongIntRec

Enter topic text here.

```
TLongIntRec = record
  case Byte of
    1: (Lo: Word;
        Hi: Word);
    2: (LoLo: Byte;
        LoHi: Byte;
        HiLo: Byte;
        HiHi: Byte);
end;
```

Defined in ogutil unit.

PCode

Enter topic text here.

PCode = ^TCode;

Defined in ogutil unit.

TCode

Enter topic text here.

```
TCode = packed record
  CheckValue : Word;              {magic value}
  Expiration : Word;              {expiration date or 0, if none}
  case Byte of
    0 : (FirstDate    : Word;      {for date code}
         EndDate      : Word);
    1 : (Days         : Word;      {for days code}
         LastAccess   : Word);
    2 : (RegString    : LongInt);  {for reg code}
    3 : (SerialNumber : LongInt);  {for serial number code}
    4 : (UsageCount   : Word;      {for usage count code}          {!!.02}
         LastChange   : Word);                                     {!!.02}
    5 : (Value        : LongInt);  {for specail codes}
    6 : (NetIndex     : LongInt);  {for net codes}
end;
```

Defined in ogutil unit.

Usable date range: 1996-Jan-02 through 2175-Jun-06.
A 0 date will be returned as 9999-Jan-1 via the ExpandDate function.

The CheckValue field is one of the following:

```
DaysCheckCode    = $649B
DateCheckCode    = $A4CB
NetCheckCode     = $9341
RegCheckCode     = $D9F6
SerialCheckCode  = $3C69
UsageCheckCode   = $F3D5
SpecialCheckCode = $9C5B
```

TCodeType

Enter topic text here.

```
TCodeType = (ctDate, ctDays, ctRegistration, ctSerialNumber,
            ctUsage, ctNetwork, ctSpecial, ctUnknown);
{order must match tab order for code generation notebook}
```

Defined in ogutil unit.

TKey

Enter topic text here.

TKey = array [0..15] of Byte;

Defined in ogutil unit.

TKeyType

Enter topic text here.

TKeyType = (ktRandom, ktMessageDigest, ktMessageDigestCS);
{order must match order for key generation combobox string list}

ktRandom	The key is generated using Delphis random number generator.
ktMessageDigest (Standard Text)	The key is generated by using the supplied text. Text case is ignored.
KtMessageDigestCS (Case-Sensitive Text)	The key is generated by using the supplied text. Text case is considered.

Defined in ogutil unit.

TTMDContext

Enter topic text here.

TTMDContext = array [0..279] of Byte;

Defined in ogutil unit.

TMD5Context

Enter topic text here.

TMD5Context = array [0..87] of Byte;

Defined in ogutil unit.

TMD5Digest

Enter topic text here.

TMD5Digest = array [0..15] of Byte;

Defined in ogutil unit.

T128Bit

Enter topic text here.

T128Bit = array [0..3] of LongInt;

Defined in ogutil unit.

T256Bit

Enter topic text here.

T256Bit = array [0..7] of LongInt;

Defined in ogutil unit.

TEsMachineInfoSet

Used to determine what factors are gathered to generate a machine identifier.

TEsMachineInfoSet = set of (midUser, midSystem, midNetwork, midDrives, midCPUID, midCPUIDJCL, midBIOS, midWinProd, midCryptoID, midNetMAC, midDomain);

Defined in ogutil unit.
Added in version 1.05.
Added in version 1.15: midCPUID, midCPUIDJCL, midBIOS, midWinProd, midCryptoID, midNetMAC, midDomain

Used by CreateMachineID function.

To maintain compatibility with version 1.13, the midUser, midSystem, midNetwork, and midDrives code has not been altered.
New factors were added instead.
Refer to the CreateMachineID function for platform specific usage.

The midCPUID factor is intended for fetching basic CPU identification.
The midCPUIDJCL factor is intended for fetching enhanced CPU identification via the JCLSysInfo routines.
The midBIOS factor is intended for fetching basic BIOS identifiers.
The midWinProd factor is intended for fetching Microsoft Windows product identifiers.
The midCryptoID factor is intended for fetching machine specific cryptography identifiers.
The midNetMAC factor is intended for fetching the MAC addresses of known network adapters.
The midDomain factor is intended for fetching the machine's domain membership.

TCodeStatus

Enter topic text here.

```
TCodeStatus = (ogValidCode,    {code is valid but may still be expired}
              ogInvalidCode,  {code is invalid}
              ogPastEndDate,  {end date has been reached}
              ogDayCountUsed, {number of days authorized have been used}
              ogRunCountUsed, {number of runs authorized have been used}
              ogNetCountUsed, {number of authorized users has been exceeded}
              ogCodeExpired); {expiration date has been reached}
```

Defined in ogutil unit.

BaseDate

BaseDate : LongInt = 0;

This is the date used as the starting point for all date fields in the TCode structure.
It is defined as a constant but set to Trunc(EncodeDate(1996, 1, 1)) in the initialization section thus requiring the Assignable Typed Constants compiler option.

Defined in ogutil unit.

OgFile.GetFileSize

Generic function to get the size of a file.
Win32 and Win64 pass through to the Windows API function GetFileSize.

Platform	Delphi	FPC / Lazarus
Win16	Yes	Yes
Win32	Yes	Yes
Win64	Yes	Yes
Linux	No	Yes
MacOS	No	
iOS	No	
Android	No	
Kylix	Yes	

FPC specific code is for non-Windows platforms

OgFile.LockFile

Support for Windows API function LockFile.

Platform	Delphi	FPC / Lazarus
Win16	Yes	Yes
Win32	No	No
Win64	No	No
Linux	No	Yes
MacOS	No	
iOS	No	
Android	No	
Kylix	Yes	

FPC specific code is for non-Windows platforms

OgFile.UnlockFile

Support for Windows API function UnlockFile.

Platform	Delphi	FPC / Lazarus
Win16	Yes	Yes
Win32	No	No
Win64	No	No
Linux	No	Yes
MacOS	No	
iOS	No	
Android	No	
Kylix	Yes	

FPC specific code is for non-Windows platforms

OgFile.FlushFileBuffers

Support for the Windows API function FlushFileBuffers.

Platform	Delphi	FPC / Lazarus
Win16	Yes	Yes
Win32	No	No
Win64	No	No
Linux	No	Yes
MacOS	No	
iOS	No	
Android	No	
Kylix	Yes	

FPC specific code is for non-Windows platforms

AutoCheck property

property TOgCodeBase.AutoCheck : Boolean

- Default: False

AutoCheck determines whether CheckCode is called automatically.

If AutoCheck is True, CheckCode is automatically called after the form containing this component is loaded. If AutoCheck is False, you are responsible for calling CheckCode to determine the component status.

See also: CheckCode

Code property

property TOgCodeBase.Code : **string**

Code is the release code.

Code is normally generated by another program, encoded using the applications key, and given to the user to enter into the application where it is decoded and validated. The behavior of the application when a code is entered is entirely up to you, the designer, and is also determined to some extent by the type of code being used.

Code is published as needed by descendent components.

See also: OnGetCode, StoreCode

Modifier property

property TOgCodeBase.Modifier : LongInt

Modifier is used to sign the key.

If Modifier is equal to 0, the key is not altered. If Modifier is not equal to 0, it is used to sign the key. Modifier is normally generated as needed, but can be stored on the stream with the form if the StoreModifier property is True.

See also: OnGetModifier, StoreModifier

StoreCode property

property TOgCodeBase.StoreCode : Boolean

- Default: False

StoreCode determines whether the release code is stored in the resource file.

StoreCode is published as needed by descendants.

See also: Code, OnGetCode

StoreModifier property

property TOgCodeBase.StoreModifier : Boolean

- Default: False

StoreModifier determines whether the modifier is stored in the resource file.

See also: Modifier, OnGetModifier

OnChecked event

property TOgCodeBase.OnChecked : TCheckedCodeEvent

TCheckedCodeEvent = **procedure**(Sender : TObject; Status : TCodeStatus) **of object**;

OnChecked defines an event handler that is called after the release code is checked.

Sender is the instance of the release code component. Status is the value returned by a call to CheckCode.

See also: CheckCode

OnGetKey event

property TOgCodeBase.OnGetKey : TGetKeyEvent

TGetKeyEvent = **procedure**(Sender : TObject; var Key : TKey) **of object**;

OnGetKey defines an event handler that is called to get the key.

Sender is the instance of the release code component.

The key should always be stored as a constant in the application and never stored in the form, a file, or the registry. Putting the key anywhere except in the application increases the chances that someone will find and be able to use it to decode the release code.

OnGetCode event

property TOgCodeBase.OnGetCode : TGetCodeEvent

TGetCodeEvent = **procedure**(Sender : TObject; var Code : TCode) **of object**;

OnGetCode defines an event handler that is called to get the release code. Sender is the instance of the release code component. Code is the TCode value associated with this component. Release codes are normally stored in a file or the registry. In some cases, the release code can be stored in the resource. To do this, set the StoreCode property to True.

See also: Code, StoreCode

OnGetModifier event

property TOgCodeBase.OnGetModifier : TGetModifierEvent

TGetModifierEvent = **procedure**(Sender : TObject; var Value : LongInt) **of object**;

OnGetModifier defines an event handler that is called to get the modifier.

Sender is the instance of the release code component. Value is the modifier that is used to sign the key. Modifier is normally generated as needed, but can be stored on the stream with the form if the StoreModifier property is True.

See also: Modifier, StoreModifier

CheckCode method

function TOgCodeBase.CheckCode(Report : Boolean) : TCodeStatus; **virtual**; **abstract**;

TCodeStatus = (ogValidCode, ogInvalidCode, ogPastEndDate, ogDayCountUsed, ogRunCountUsed, ogNetCountUsed, ogCodeExpired);

CheckCode checks for a valid release code.

CheckCode is defined as virtual and abstract, which means that each descendant component overrides it to provide the necessary code to validate and test the release code obtained through the Code property. If Report is True, the result of the test is reported by triggering the OnChecked event. If Report is False, you must check the function result.

CheckCode requires several pieces of information, which it obtains by triggering event handlers that you define. The normal sequence of events performed by CheckCode is:

1. Trigger the OnGetKey event to get the key used to encode and decode the release code. The key should always be embedded in the application as a constant.
2. Trigger the OnGetCode event to get the release code. The release code is normally stored in the registry or an INI file.
3. Trigger the OnGetModifier event to get the key modifier. The modifier can be stored as a constant in the application, stored in the registry or INI file, or generated when it is needed.
4. Apply the modifier to the key.
5. Test the release code to see if it is valid.
6. Test the release code to see if it has expired. The details of this test depend on the type of release code.

The result of calling CheckCode is one of the following values:

ogValidCode	release code is valid.
ogInvalidCode	release code is invalid (the internal integrity check failed).
ogPastEndDate	ending date has past.

ogDayCountUsed	authorized days have been used.
ogRunCountUsed	authorized runs have been used.
ogNetCountUsed	number of authorized users has been exceeded.
ogCodeExpired	The expiration date has been reached.

See also: AutoCheck, OnChecked, OnGetCode, OnGetKey, OnGetModifier

IsCodeValid method

function TOgCodeBase.IsCodeValid : Boolean;

IsCodeValid tests to see if the release code is valid.

IsCodeValid calls the CheckCode method and tests its result to see if the release code is valid. It returns True if the code is valid, otherwise False. Descendent components decode the release code and test to see if the signature value (the magic value as defined in the TCode record) is still valid.

You might need to perform additional tests to ensure that the data used to create the release code was not altered. For example, you could test whether the text string used to create a Simple Registration release code was altered. Since the string is not part of the release code (only a number derived from the string is embedded into the code), you cannot compare it to what is stored in the release code. You must create a temporary release code using the text string and the same expiration date and then compare the temporary release code to the stored one. If they dont match, someone has altered the text string.

See also: CheckCode

GetValue method

function TOgDateCode.GetValue : TDateTime;

GetValue returns the end date embedded in the release code. The returned value is a Delphi TDateTime value.

AutoDecrease property

property TOgDaysCode.AutoDecrease : Boolean

- Default: True

AutoDecrease determines whether the day count value is automatically decreased each day the application is run.

If AutoDecrease is True, the day count embedded in the release code is automatically decreased by one each day the application is run. This is accomplished by calling the Decrease method. If AutoDecrease is False, you must call the Decrease method manually whenever necessary.

See also: Decrease

OnChangeCode event

property OnChangeCode : TChangeCodeEvent

TChangeCodeEvent = **procedure**(Sender : TObject; Code : TCode) **of object**;

OnChangeCode defines an event handler that is called when a release code changes.

This event is fired after the release code is changed via a call to Decrease, either directly or automatically (if the AutoDecrease property is True).

Sender is the instance of the release code component. Code is the new release code value.

The release code should be saved in an INI file or the registry.

See also: AutoDecrease, Decrease

Decrease method

procedure TOgDaysCode.Decrease;

Decrease reduces the day count value stored in the release code.

Performing this action requires several vital pieces of information, which are normally obtained by triggering several event handlers that you define. The normal sequence of events performed by Decrease is:

1. Trigger the OnGetKey event to get the key used to encode and decode the release code. The key should always be embedded in the application as a constant.
2. Trigger the OnGetCode event to get the release code. The release code is normally stored in the registry or an INI file.
3. Trigger the OnGetModifier event to get the key modifier. The modifier can be stored as a constant in the application, stored in the registry or INI file, or generated when it is needed.
4. Apply the modifier to the key.
5. Test the code to see if it is valid.
6. Decrease the day count by one if it has not already been decreased today.
7. Trigger the OnChangeCode event to store the changed release code.

See also: OnChangeCode, OnGetCode, OnGetKey, OnGetModifier

GetValue method

function TOgDaysCode.GetValue : LongInt;

GetValue returns the day count embedded in the release code. The value returned is the number of days remaining.

Code property

property TOgMakeCodes.Code : TCode

Code is the generated release code.

After a successful call to Execute, Code contains the generated release code.

Code can represent any one of several release code types. Use the CodeType property to determine which code type was generated.

See also: CodeType, Execute

CodeType property

property TOgMakeCodes.CodeType : TCodeType

CodeType is the type of release code.

If you assign a value to CodeType prior to calling Execute, the corresponding notebook page is displayed in the Code Generation dialog. After a successful call to Execute, CodeType contains the type of code that was generated. The ctUnknown code type is only used internally. The default is ctDate.

See also: Execute

Key run-time property

property TOgMakeCodes.Key : TKey

Key is used to encode and decode the release code.

The key used to encode release codes should be protected from unauthorized use because a release code that was encoded without a modifier can easily be decoded using the key.

The key should be embedded into the application rather than stored in a file or resource.

If no value is assigned to this property, the Key Maintenance dialog is displayed so that a key can be selected or created.

See also: Code

KeyFileName property

property TOgMakeCodes.KeyFileName : **string**

KeyFileName is the name of the INI file used to store application names and their associated keys.

If a valid file name is assigned to this property, its contents are displayed when the Key Maintenance dialog is displayed.

KeyType run-time property

property TOgMakeCodes.KeyType : TKeyType

TKeyType = (ktRandom, ktMessageDigest, ktMessageDigestCS);

- Default: ktMessageDigest

KeyType determines the type of key to generate.

The valid key types are:

ktRandom	The key is generated using Delphis random number generator.
ktMessageDigest (Standard Text)	The key is generated by using the supplied text. Text case is ignored.
KtMessageDigestCS (Case-Sensitive Text)	The key is generated by using the supplied text. Text case is considered.

If a value is assigned to this property, it is used to determine the type of key to generate when the Key Maintenance dialog is displayed.

ShowHints property

property TOgMakeCodes.ShowHints : Boolean

- Default: False

ShowHints determines whether hints are shown for the TOgMakeCodes dialogs.

Execute method

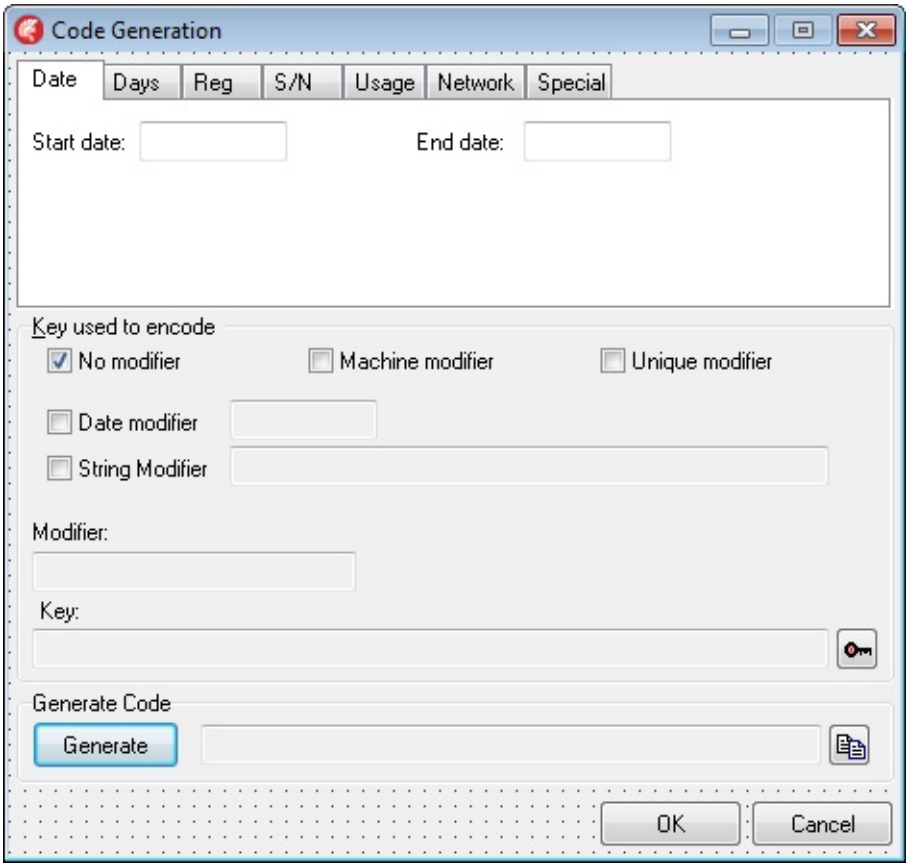
function TOgMakeCodes.Execute : Boolean;

Execute displays the Code Generation dialog.

Use this method to display the Code Generation dialog so that a release code can be generated.

If Execute returns True, the Code and CodeType properties contain valid values. Otherwise, the contents of these properties is unknown.

See also: Code, CodeType



Date | Days | Reg | S/N | Usage | Network | Special

Day count: 0 Expires:

Date | Days | Reg | S/N | Usage | Network | Special

String:

Random Number Expires:

Date | Days | Reg | S/N | Usage | Network | Special

Serial Number: 0 Expires:

Random Number

Date | Days | Reg | S/N | Usage | Network | Special

Usage count: 0 Expires:

Date | Days | Reg | S/N | Usage | Network | Special

Access Slots: 2

Date | Days | Reg | S/N | Usage | Network | Special

Special data: 0 Expires:

Key run-time property

property TOgMakeKeys.Key : TKey

TKey = array [0..15] of Byte;

Key is used to encode and decode release codes.

After a successful call to Execute, Key contains the selected key value.

The key used to encode release codes should be protected from unauthorized use because a release code that was encoded without a modifier can easily be decoded using the key.

The key should be embedded into the application rather than stored in a file or resource.

See also: ApplyModifierToKey, Execute, GenerateKey, GenerateRandomKey

KeyFileName property

property TOgMakeKeys.KeyFileName : **string**

KeyFileName is the name of the INI file used to store application names and their associated keys.

If a valid file name is assigned to this property, its contents are displayed when the Key Maintenance dialog is displayed.

KeyType property

property TOgMakeKeys.KeyType : TKeyType

TKeyType = (ktRandom, ktMessageDigest, ktMessageDigestCS);

- Default: ktMessageDigest

KeyType determines the type of key to generate.

After a successful call to Execute, KeyType contains one of these key types.

ktRandom	The key is generated using Delphis random number generator.
ktMessageDigest (Standard Text)	The key is generated by using the supplied text. Text case is ignored.
ktMessageDigestCS (Case-Sensitive Text)	The key is generated by using the supplied text. Text case is considered.

If a value is assigned to this property, it is used to determine the type of key to generate when the Key Maintenance dialog is displayed.

See also: Execute

ShowHints property

property TOgMakeKeys.ShowHints : Boolean

- Default: False

ShowHints determines whether hints are shown for the TOgMakeKeys dialogs.

ApplyModifierToKey method

procedure TOgMakeKeys.ApplyModifierToKey (Modifier : LongInt; var Key; KeySize : Cardinal);

ApplyModifierToKey alters the specified key.

If Modifier is not zero, this routine alters (signs) the key specified by Key. KeySize is the size, in bytes, of Key .

This routine is used automatically by the components that generate a release code when a non-zero value is specified for the Modifier property.

See also: GenerateDateModifier, GenerateMachineModifier, GenerateStringModifier, GenerateUniqueModifier, Key

Execute method

function TOgMakeKeys.Execute: Boolean;

Execute displays the Key Maintenance dialog.

Use this method to display the Key Maintenance dialog so that a key can be generated. The dialog is described in the "Creating and Maintaining Keys" section of the manual.

If Execute returns True, the Key, KeyFileName, and KeyType properties contain valid values. Otherwise, the contents of these properties is unknown.

See also: Key, KeyFileName, KeyType

GenerateDateModifier method

function TOgMakeKeys.GenerateDateModifier: LongInt;

GenerateDateModifier creates a key modifier based on the current date.

This routine is also available as a function (GenerateDateModifierPrim) for use in applications that need to generate modifiers dynamically.

See also: ApplyModifierToKey, GenerateMachineModifier, GenerateStringModifier, GenerateUniqueModifier

GenerateKey method

procedure TOgMakeKeys.GenerateKey (var Key; KeySize : Cardinal; const Str : **string**);

GenerateKey produces a key based on a supplied text string.

To produce keys that are not case dependent, convert the text to upper case prior to calling GenerateKey.

See also: ApplyModifierToKey, GenerateRandomKey, Key

GenerateMachineModifier method

function TOgMakeKeys.GenerateMachineModifier: LongInt;

GenerateMachineModifier creates a key modifier based on the hardware information for the current machine.

GenerateMachineModifier uses hard disk volume sizes, volume serial numbers, registration name/company as reported by Windows, and the network card ID (if available) to produce a modifier specific to a single machine.

Use this modifier to sign the key used to encode and decode release codes if you want the release code to restrict usage to a single machine.

Caution: If hardware is changed on the machine, the modifier changes, rendering the release code, and consequently the application, unusable.

This routine is also available as a function (GenerateMachineModifierPrim) for use in applications that need to generate modifiers dynamically.

See also: ApplyModifierToKey, GenerateDateModifier, GenerateStringModifier, GenerateUniqueModifier

GenerateRandomKey method

procedure TOgMakeKeys.GenerateRandomKey(var Key; KeySize : Cardinal);

GenerateRandomKey produces a key based on Delphis internal random number generator.

See also: ApplyModifierToKey, GenerateKey, Key

GenerateStringModifier method

function TOgMakeKeys.GenerateStringModifier (const S : **string**) : LongInt;

GenerateStringModifier creates a key modifier based on the supplied string.

This routine is also available as a function (GenerateStringModifierPrim) for use in applications that need to generate modifiers dynamically.

See also: ApplyModifierToKey, GenerateDateModifier, GenerateMachineModifier, GenerateUniqueModifier

GenerateUniqueModifier method

function TOgMakeKeys.GenerateUniqueModifier: LongInt;

GenerateUniqueModifier creates a unique key modifier.

This routine is also available as a function (GenerateUniqueModifierPrim) for use in applications that need to generate modifiers dynamically.

See also: ApplyModifierToKey, GenerateDateModifier, GenerateMachineModifier, GenerateStringModifier

ActiveUsers read-only property

property TOgNetCode.ActiveUsers : LongInt

ActiveUsers is the current number of users running the application.

FileName property

property TOgNetCode.FileName : **string**

FileName is the name of the Network Access File.

The Network Access File is used to determine if another instance of the application is authorized. If the file specified in FileName does not exist, it is created and initialized during the call to CheckCode.

InvalidUsers read-only property

property TOgNetCode.InvalidUsers : LongInt

InvalidUsers is the number of invalid user access slots in the Network Access File.

Invalid slots are created when the user does not exit the application normally. Use ResetAccessFile to fix these invalid slots.

See also: ResetAccessFile

MaxUsers read-only property

property TOgNetCode.MaxUsers : LongInt

MaxUsers is the maximum number of concurrent users of the application.

IsRemoteDrive method

function TOgNetCode.IsRemoteDrive(const ExePath : **string)**
: Boolean;

IsRemoteDrive determines whether ExePath resides on a remote disk drive.

You can use IsRemoteDrive to determine if your application is being run from a remote disk drive. Only the drive information passed in ExePath is used.

ResetAccessFile method

function TOgNetCode.ResetAccessFile : Boolean;

ResetAccessFile resets the invalid slots in the Network Access File.

If the operation is successful, the return value is True. If the file could not be opened for write access, the return value is False.

Calling ResetAccessFile does not effect active users. Since their access slots are in use, they are assumed to be valid and are not reset.

AutoCheck property

property TOgProtectExe.AutoCheck : Boolean

- Default: False

AutoCheck determines whether CheckExe is called automatically.

If AutoCheck is True, CheckExe is called after the form containing this component is loaded. If AutoCheck is False, you are responsible for calling CheckExe to determine the status of the executable file.

See also: CheckExe

CheckSize property

property TOgProtectExe.CheckSize : Boolean

- Default: True

CheckSize determines whether the size of the executable is tested.

If CheckSize is True, the size and the CRC of the executable file are tested. If CheckSize is False, only the CRC of the executable file is tested.

OnChecked event

property TOgProtectExe.OnChecked : TCheckedExeEvent

TCheckedExeEvent = **procedure**(Sender : TObject; Status : TExeStatus) **of object**;

OnChecked defines an event handler that is called after the executable is checked.

Sender is the instance of the release code component. Status is the value returned by a call to CheckExe.

See also: CheckExe

CheckExe method

function TOgProtectExe.CheckExe(Report : Boolean) : TExeStatus;

TExeStatus = (exeSuccess, exeSizeError, exeIntegrityError, exeNotStamped);

CheckExe tests to see if the executable file was altered.

If Report is True, the result of the test is reported by triggering the OnChecked event. If Report is False, you must check the function result.

The result of calling CheckExe is one of the following values:

exeSuccess	executable file has not changed.
exeSizeError	size of the executable file changed.
exeIntegrityError	or more bytes in the executable changed.
exeNotStamped	The executable is not stamped with the CRC and size information.

See also: OnChecked

StampExe method

function TOgProtectExe.StampExe (const FileName : **string** ; EraseMarker : Boolean) : Boolean;

StampExe marks the executable program with its size and a CRC value.

StampExe searches for a special marker that is used to mark the record where the size and CRC value are stored, calculates the executables size and CRC, and writes that information back to the record. If EraseMarker is True, the special marker used to locate the record is erased.

This method is not used by the TOgProtectExe component. It is provided so that you can use it to stamp the application you want to protect. You can write a simple application that uses StampExe to stamp the application you want to protect. Or you can use the STAMPEXE example project (which uses the StampExe method) to stamp the application you want to protect.

See also: UnStampExe

UnStampExe method

function TOgProtectExe.UnStampExe (const FileName : **string**) : Boolean;

UnStampExe reverses the effect of a call to StampExe.

UnStampExe can only be used if the special marker used to locate the CRC record was not erased by StampExe.

This method is not used by the TOgProtectExe component. It is provided so that you can use it unstamp an application.

See also: StampExe

RegString property

property TOgRegistrationCode.RegString : **string**

RegString is the registration string used to create the release code.

See also: OnGetRegString

StoreRegString property

property TOgRegistrationCode.StoreRegString : Boolean

- Default: True

StoreRegString determines whether the registration string value is stored as a resource at design time.

If StoreRegString is True, the value of RegString is stored in the resource file along with the form. If StoreRegString is False, RegString is not stored and you must supply an OnGetRegString event handler so that the registration string can be retrieved when required.

See also: OnGetRegString, RegString

OnGetRegString event

property TOgRegistrationCode.OnGetRegString : TGetRegStringEvent

TGetRegStringEvent = **procedure**(Sender : TObject; var Value : **string**) **of object**;

OnGetRegString defines an event handler that is called to get the registration string.

Sender is the instance of the release code component. Value is the registration string used to create the release code.

GetValue method

function TOgSerialNumberCode.GetValue : LongInt;

GetValue returns the serial number embedded in the release code.

The value returned is the serial number that was used when the release code was created.

GetValue method

function TOgSpecialCode.GetValue : LongInt;

GetValue returns the special information embedded in the release code.

The returned value is a LongInt. The interpretation of the returned value is determined entirely by you.

AutoDecrease property

property TOgUsageCode.AutoDecrease : Boolean

- Default: True

AutoDecrease determines whether the usage count value is automatically decreased each time the application is run.

If AutoDecrease is True, the usage count value embedded in the release code is automatically decreased by one each time the application is run. When the usage count is reduced to zero, the release code is expired. If AutoDecrease is False, you must call the Decrease method manually whenever necessary.

See also: Decrease

OnChangeCode event

property TOgUsageCode.OnChangeCode : TChangeCodeEvent

TChangeCodeEvent = **procedure**(Sender : TObject; Code : TCode) **of object**;

OnChangeCode defines an event handler that is called when a release code changes.

This event is fired after the release code is changed via a call to Decrease, either directly or automatically (if the AutoDecrease property is True).

Sender is the instance of the release code component. Code is the new release code value.

The release code should be saved in the INI file or the registry.

See also: AutoDecrease, Decrease

Decrease method

procedure TOgUsageCode.Decrease;

Decrease reduces the usage count value stored in the release code.

Performing this action requires several vital pieces of information, which are normally obtained by triggering several event handlers that you define. The normal sequence of events performed by Decrease is:

1. Trigger the OnGetKey event to get the key used to encode and decode the release code. The key should always be embedded in the application as a constant.
2. Trigger the OnGetCode event to get the release code. The code is normally stored in the registry or an INI file.
3. Trigger the OnGetModifier event to get the key modifier. The key modifier can be stored as a constant in the application, stored in the registry or INI file, or generated when it is needed.
4. Apply the modifier to the key.
5. Test the release code to see if it is valid.
6. Decrease the usage count by one.
7. Trigger the OnChangeCode event to store the changed release code.

See also: OnChangeCode, OnGetCode, OnGetKey, OnGetModifier,

GetValue method

function TOgUsageCode.GetValue : LongInt;

GetValue returns the usage count embedded in the release code. The value returned is the number of runs remaining.