DEC version history since V5.2

Document version: 1.0 as of 14th February 2021

Table of contents

1 Overview of changes made between DEC 6.0 and 6.1

This chapter describes the most important changes made between DEC 6.0 and DEC 6.1. It is only important for users already having used DEC 6.0 and updating to DEC 6.1 now.

!!! This is currently an alpha version so everything may still change !!!

The main changes are:

- Replaced the *IDECProgress* interface by an anonymous method. Details see chapter 3.39 in the main documentation.
- Changed the behaviour of the *Size* parameter for *CalcStream* methods. It always starts from the current position of the stream now.
- A new *VCL* based demo program *ProgressDemoVCL* has been added to demo the use of the *TDECProgressEvent* mechanism for getting information about the process of lengthy operations.
- Add the new formats *TFormat_BigEndian16*, *TFormat_BigEndian32* and *TFormat_BigEndian64*.
- Add a new console application for adding the library paths to the RAD Studio IDE settings when performing a manual installation and a batch file for compiling everything in one step.
- The *EDECAbstractError* exception constructor parameter type changed to be able to get rid of a unit uses cycle involving *DECUtil.pas* and *DECBaseClass.pas*.
- Some internal structure changes, which should not affect ordinary DEC users.
- The *VCL_CipherWorkbench* demo got removed, as this was not really implemented anyway.

- Added the SHA224 hash algorithm which is part of the SHA-2 family of algorithms.
- Added HMAC (hash message authentication code) algorithm (rfc2202).
- Added PBKDF2 (password based key deviation function 2) algorithm (rfc2898, PKCS #5).
- Renamed `Protect` method to `SecureErase` in `TCipherBase` class and made it protected instead of public.
- Introduced a new class `TDECHashAuthentication` in a new unit `DECHashAuthentication` and moved all KDF, MGF, HMAC and PBKDF2 algorithms in this new class. Made all hash implementations inherit from this one. Also moved the IsPasswordHash to this new class.
- Fixed regressions introduced in the following ciphers, which no longer passed their unit tests on x64 platform: Blowfish, RC6 and Q128. We apologize for having them released in a broken state.
- Fixed a regression in Sapphire cipher which worked for some data like it is contained in the unit tests but a user found data and problematic code where this failed.

2 Overview of changes made between DEC 5.2 and 6.0

This chapter describes the most important changes made between DEC 5.2 and DEC 6.0. It is only important for users already having used DEC 5.2 and updating to DEC 6.0 now.

The main changes are:

- The names of all units have been changed to more verbose ones making them clearer. New units have been added to better structure the code. In some cases, things have been divided in a base unit, implementing base functionality shared by all algorithms and defining the basic public API of the individual implementing classes, and another unit with the actual implementations of the actual algorithms. This is the case for the formatting routines, the ciphers and the hashes.

  In addition to changing names the *DECData* unit has been split up into three units: *DECData*, which contains all constants used by both cipher- and hash algorithms, *DECDataCipher* (containing constants only relevant for cipher algorithms) and *DECDataHash* (containing all constants which are only relevant for hash algorithms). This has been done in order to increase modularity and for not including anything not relevant for a cipher- or hash algorithm only user.

- The directory structure has been changed with DEC_Part_II having been removed. That was a DCU only compilation of some advanced algorithms coded by the original author of DEC. But only DCUs for Delphi 7 were shipped, so it was not of use for most DEC users anymore. We do not have the code of these units.

- The use of assembler within the library has been made optional and it also has been cut back at a few places where it really did not bring much if any speed improvements. If you want to turn on

the use of assembler you have to change the *NO_ASM* conditional define in the *DECOptions.inc* include file. Be aware that you lose cross platform compatibility (even with 64 bit Windows!) by doing this.

- All uses of *PAnsiChar* have been replaced, since that data type is not available on mobile compilers and most likely will never be.

- Various methods using *TBytes* as buffer for binary data have been introduced. Since Delphi XE7 there is improved support for handling dynamic arrays e.g. by providing support for *Insert*, *Delete* and *Concat* on them. Now there is no longer any need to use string like data types for storing binary data. Back in the ANSI days it was simply convenient, as ANSI based strings provided this functionality usually without data loss, but only when being used with the same code page.

- All methods which used the data type *Binary* have been changed to directly use *RawByteString* and their names also changed to reflect this. But at the same time, they have been deprecated in favour to the *TBytes* oriented methods.

- Some more CRC variants have been added after looking up whether the already commented out data was correct.

- Some basic Demos have been added.

- DUnit tests are now being provided for the formatting classes, the CRC-, hash- and crypto-algorithms and to a lesser extend (due to its nature) for the random number generator as well.

- The register method used when creating non-visual components based on DEC has been removed as we decided that support for components would be too much of a hassle.

- A way to easily get the exception messages translated on Firemonkey mobile projects has been implemented. For using it you must enable the *FMXTranslateableExceptions* define in *DECOptions.inc*.

- A new class method *IsPasswordHash* has been introduced to the hash classes. Currently this will always return false as we do not have any classes which inherit from the new *TDECPasswordHash* class yet. But in future versions this can be used to easily find out if a certain hashing class is particular designed for hashing passwords.

- The Cipher context record got a new field *CipherType* added. This is a set and can be used for easily finding out some properties about the cipher algorithm like whether this is a block or a stream-oriented algorithm, whether it is symmetric or asymmetric. Be aware that DEC 6.0 doesn't contain asymmetric ciphers yet.

- The Cipher context record got two additional new fields added *MinRounds* and *MaxRounds*. For cipher algorithms having a *Rounds* property, defining the number of iterations the algorithm performs over the data, these two properties specify the minimum and maximum

values for the *Rounds* property. The setter of the property of course will enforce these values, but they can be useful for general encryption applications where you may choose the algorithm used and its properties. For algorithms not having a *Rounds* property *MinRounds* and *MaxRounds* will both return 1. There are a few algorithms using rounds internally but defining the number of rounds by some algorithm. For those *MinRounds* and *MaxRounds* will both return 1 either as for those the user cannot change the *Rounds* value anyway.

- The format *TFormat_MIME32* has been renamed to *TFormat_DECMIME32* to make it clearer that this is a DEC specific format. An alias with the old class name is being provided but in general we do not recommend using such a DEC specific formatting.

- The format *TFormat_MIME64* has been renamed to *TFormat_BASE64*, as this is the more standard name of this format. An alias with the old class name is being provided, but marked as deprecated to encourage you to switch to the new name.

- The class registration mechanism has been reworked.

- A common interface *IDECCipher* useable for all cipher algorithms has been introduced to enable interface based programming.

- The *THash_SHA* class has been renamed to *THash_SHA0* to make it more clear which hash algorithm this is. For backwards compatibility (but remember: you really should not use the SHA0 algorithm due to security issues), a define has been introduced: *OLD_SHA_NAME*. If this is enabled in *DECOptions.inc* a *THash_SHA* class directly descending from and doing the very same thing as the *THash_SHA0* class is declared. Be aware that the *THash_SHA0* class has a different *identity* value than the *THash_SHA* class. The identity concept is described in chapter 3.4 in the main documentation.

- The *THash_Whirlpool* class has been renamed to *THash_Whirlpool0* and *THash_Whirlpool* class has been renamed to *THash_Whirlpool1*. For backwards compatibility, in case you need the old class names or identity values, a define has been introduced: *OLD_WHIRLPOOL_NAMES*. If this is enabled in *DECOptions.inc* the old names *THash_Whirlpool* and *THash_Whirlpool* are being defined as well. The current variant of the Whirlpool hash has been implemented as *THash_Whirlpool1* or if the *OLD_WHIRLPOOL_NAMES* define is being used as *THash_Whirlpool1New*. It is recommended to use that variant if the code does not need to be backwards compatible.

- It has been found out that the *TDEC_Hash.KDF2* implementation actually was a *KDF1* implementation as it has been used by the MGF1 method, which uses KDF1 according to the references found in the internet. The complete KDF implementation (except for the KDFx and MGFx implementations) has been reworked and KDF1, KDF2 and KDF3 have been implemented now. So KDF2 and KDF3 are new in DEC 6.0 and KDF1 is properly surfaced as KDF1 now. Some unit tests for all three variants have been added as well.

If you formerly used one of the *KDF2* methods, you need to change this to call the corresponding *KDF1* method now!

- A define `ManualRegisterClasses` has been introduced on request of a user. It seems this had already been in DEC 3.0 but got removed/lost later on. When being defined (the default is off) no formatting-, hash- or cipher-classes are being automatically registered in the initialization sections of the corresponding units. These classes do not need to be registered for use. But if they are not registered one cannot use the class registration mechanism which can be handy for getting an instance of an algorithm specified by ID in a file.

- Changed `DigestSize` and `BlockSize` of the hash-classes from *Integer* to *UInt32* as a signed data type does not make much sense there.

- Changed name of `Snefru SecurityLevel` property to `Rounds` to make it more uniform to the other hash classes already offering a rounds property. In addition, this property defined the number of cycles (rounds) for the algorithm anyway so the new name is correct as well.

- The XTEA cipher's DoEncode/DoDecode methods have been changed to match the C implementations commonly found on the internet. The brackets in DEC were at wrong places or missing and thus the output was different.

If you formerly used the XTEA cipher the new version will provide different results! The old implementation of these two methods can be found in DEC V5.2.

- The maximum number of rounds allowed for the TEA and XTEA block ciphers were changed from 32 to 256 as the recommended value is already 64 rounds according to Wikipedia.