DEC version history since V5.2

Document version: 1.7 as of 23rd December 2024

Table of contents

DEC 6.5 requires Delphi 10.1 Berlin as minimum version.

1 Overview of changes made between DEC 6.5 and DEC 6.4.1

New algorithms:

- Added the BCrypt password hashing algorithm class *THash_BCrypt* and new functionality for Crypt/BSD style password management.
- Added a generic mechanism to calculate Crypt/BSD compatible output for password hashes which are Crypt/BSD compatible and implemented this scheme for BCrypt.
- Added a Crypt/BSD style output format for the BCrypt algorithm, called *TFormat_BCryptBSD*.
- Added a Base32 format class *TFormat_Base32*.
- Added a UTF8 format class *TFormat_UTF8*.
- Added a UTF16 format class *TFormat_UTF16*. This is an Alias of *TFormat_BigEndian16*.
- Added Keccack hash algorithm, which later got SHA3 after modifying the padding sheme. If other libraries calculated hash values based on Keccack rather than SHA3 (some implementations didn't seem to note the slight difference in the implementation of the padding) one can produce the same outputs as those now.
- Added possibility to pad the data to encrypt using the PKCS#7 algorithm as specified in RFC 5652. For this *TDECCipher.FillMode* got renamed into *TDECCipher.PaddingMode*.

Bugfixes:

- Fixed raising of an exception in *CalcFile* and *CalcStream* methods when being used for any of the SHA3 algorithms.
- Fixed a bug in *Hash_FMX* demo, which led to not properly calculating the hash values for the SHA3 algorithms when an incomplete last byte should be used. In that case the if responsible for deciding whether to pass the value of such a byte was wrong.
- Fixed the destructor of the hash base class. It should properly overwrite the internal buffer now.
- Fixed a bug in *DECOptions.inc* which prevented using pure pascal implementations, even when *NO_ASM* define was enabled explicitly.
- Added the missing restoration of range and overflow checks at the end of *DECHash.pas*
- Fixed authentication tag issues in GCM block concatenation mode when being used with streams
- Fixed a bug in PBKDF2 algorithm which led to wrong output beyond the first block size bytes
- Fixed a bug in GCM authentication data generation for large data
- Fixed missing class name for the optional DecodeCTS3 method (for the deprecated CTS3 padding scheme)
- Fixed wrong file names (was …VLC instead of …VCL) in ProgressDemo sample project
- Fixed TDECCipher.DecodeRawByteString for some cases
- Do not raise exceptions for empty keys if the cipher algorithm selected does not use keys or allows for empty ones (ROT13, which is not implemented in DEC and GCM come to mind).

Demos:

- Added a mechanism to *HashBenchmark FMX* demo to detect that an algorithm is a password hash algorithm and thus needs a salt value and adapted the size of the data processed in such cases as otherwise runtime would be extraordinarily high.
- Added a mechanism to *Hash FMX* demo to detect whether the selected algorithm is a password hash algorithm and enables support for entering the salt value. If the selected algorithm is BCrypt, support for defining the cost factor is provided as well.
- Added *Password_Console* demo project to demonstrate use of the BCrypt algorithm and the Crypt/BSD like convenience methods.
- Added *RandomComparison_VCL* demo. This compares DEC's pseudo random number generator with that of the RTL.
- Fixed wrong XMLDOC comment of IFiller parameter for TDECCipher.Init methods.
- Enhanced *Progress_VCL* and *Cipher_FMX demos*. They allow generation of initialization vector now and support the new PKCS7 padding sheme.

Other changes/improvements:

- Added an additional overload for CalcStream which does not call Init and only calls Done when the new *DoFinalize* parameter is passed as *true*. This allows to add data to the stream several times before calculating the hash value.
- Added specific classes for AES128, 192 and 256. Up to now all of these were automatically chosen depending on the key size used for *TCipher_AES*. These new classes can now even be used in cases where a too short key had been used.
- Added class types for all intermediate classes like *TDECHashAuthentication* or *TDECPasswordHash*.
- Introduced a new *TDECHashExtended* class into which all those hash calculation methods got moved which are not suitable for the password hash algorithms, specifically the ones operating on streams and files. It inherits from *TDECPasswordHash*.

- Added a method which tests whether a given password belongs to a given Crypt/BSD password database record. Or in short: is a valid password for this record.
- Added a *ClassByCryptIdentity* class method to *TDECPasswordHash*. This can be used to find the class type in the class registration list just by specifying the Crypt/BSD ID (Crypt identity) that the algorithm uses. With the class type an actual instance can be generated.
- *TFormat_HEX.DoDecode* and *TFormat_UU.Decode* throw an *EDECFormatException* instead of an *EDECException* now, if data is passed in a wrong format.
- Added a *FilterChars* method to the formatting classes which returns the chars allowed as input
- Improved XMLDOC by adding a special tag for exceptions thrown. This just might not be properly displayed in the HelpInsight popups of some Delphi versions. In 11.1 it will be shown though.
- Started to remove/replace some code constructs necessary to support versions of Delphi older than 10.1. In some cases, this will provide some small speed gains.
- Improved FPC 3.x support
- Added a DECZIPHelper unit to make it easier to implement support for creating and unzipping of encrypted ZIP files
- Replaced the *PByteArray* data type with some self-defined one to allow for bigger data blocks to be processed

2 Overview of changes made between DEC 6.4.1 and DEC 6.4

- Fixed bugs which prevented compilation on older versions, especially on XE2.
- Reworked Cipher FMX demo. Selecting an authenticated cipher mode will be detected reliably now, the order of controls got rearranged and a copy to clipboard button was added. Also a success message for a correctly calculated authentication value was added when an expected authentication value was entered by the user.
- Fixed bugs surfacing at least in 2DES algorithm when using them with cmCBCx block chaining mode. The bugs had been introduced by the cmGCM related work. Added a regression test using the test data provided by the reporter of the bug.
- Added *Cipher_Console_KDF* demo application.

3 Overview of changes made between DEC 6.4 and DEC 6.3

- Fixed bugs in *ModuleUnload* procedure of *DECHashBase.pas, DECCipherBase.pas* and *DECFormatBase.pas*. By fixing these issues C++ Builder compatibility of the codebase increased as a crash at the end of the test application used to test some C++ Builder related things no longer happens.
- Tested all the commented out 32 bit asm hash implementations (see *DECOptions.inc*) for C++ Builder compatibility (using the Clang compiler variant). This resulted in *Snefru128* and *Snefru256* no longer being commented out.

- The *Rounds* property of the *TCipher_Cast128* cipher class has been removed, as per rfc2144 the number of rounds strictly depends on the key length given and so the initialization of the rounds has been changed to comply with this this in every case.
- The minimum value for the *Rounds* property of *TCipher_RC5* has been changed from 1 to 0, as rfc2040 defines it.
- Fixed a crash in *TDECHash.CalcStream*, which occurred when this was used for one of the algorithms which can deal with partial last bytes (SHA3) and size parameter was less than zero.
- Added support for the GCM (Galois Counter Mode) block chaining mode.
- Increased test coverage by using a code coverage tool to identify missing unit tests.
- Fixed support for using TestInsight IDE plugin with the DUnit test project. Added chapter to documentation describing its use.
- Fixed layout issues in Hash FMX demo and updated the Google Play release of it
- Added GCM specific fields to Cipher FMX demo
- Revised documentation.

4 Overview of changes made between DEC 6.3 and DEC 6.2.1

- Fixed a failure in *THash_SHA3Base.Digest* which had made this method unreliable.
- Implemented Shake128 and Shake256 SHA3 algorithm variants. They contain the possibility to specify output length of the resulting hash.
- Implemented an interface for variable output length hash algorithm implementations.
- Implemented an interface for the Hash classes exposing a *Rounds* property and changed the *GetMinRounds*/*GetMaxRounds* methods to normal methods to be able to expose them in this interface as well. Up to now they were class methods.
- Added GUUIDs to all interfaces.
- Extended the *Hash_FMX* demo application by adding the possibility to change rounds and number of bits used of the final byte for algorithms supporting the respective interfaces.
- Splitted *ManualRegisterClasses* defines into these three defines: *ManualRegisterFormatClasses, ManualRegisterCipherClasses, ManualRegisterHashClasses* to be able to differentiate between those.
- Fixed bug in *Cipher_FMX* demo, where decryption did not properly work.
- Updated those 32 bit asm hash implementations (MD2, SHA384, Tiger, Snefru128 and Snefru256), which were commented out due to C++ Builder issues (as per comment from the former developers), to use the renamed field names and thus made them useable again.
- Enabled the aforementioned asm hash implementations for Delphi, for C++ Builder they remain deactivated until we could verify they work. C++ Builder programs should either be compiled from the IDE or when compiling from command line using the -DBCB parameter so DEC can detect that it's compiled by C++ Builder and then deactivate those implementations.
- Small preparations for potential GCM implementation in the Cipher unit tests.
- Implemented unit tests for setting *Rounds* for those hash classes having a *Rounds* property.
- Setting *Rounds* for any variant of the Haval hashes now ensures that it is at least the value of *GetMinRounds* for the used Haval variant. Before it was a fixed value of 3.
- In the *Hash_FMX* demo an issue when running on Windows has been fixed. The scrollbar overlapped other controls.

- Fixed the use of a wrong class in *ModuleUnload* procedure of *DECHashBase.pas*.
- The exception classes and the progress event types have been moved from *DECUtils.pas* to *DECTypes.pas*. Code using those types might need to be updated.

5 Overview of changes made between DEC 6.2.1 and DEC 6.2

- Fixed a failure introduced by version control in the deployproj file of *Hash_FMX* Demo which prevented proper loading of this demo
- Fixed FMX *TStringGrid* issues with older Delphi versions which do not have the *FMX.Grid.Style* unit. These Delphi versions can compile and run the *Cipher_FMX* and *HashBenchmark_FMX* demos now. For the benchmark demo one column contents is just not right aligned for these versions.
- Fixed use of non-existing *IFMXDialogServiceAsync* interface for versions older than 10.1 Berlin.

While we implemented these fixes for older versions we do not guarantee compatibility of every demo etc. with older versions and it is very likely that a future version will require a newer Delphi version as minimum supported version for DEC anyway! We strongly suggest to upgrade to a newer Delphi version!

6 Overview of changes made between DEC 6.1.1 and DEC 6.2

- Enhanced the *IDECCipher* interface by adding the Mode property, which can be used to set the block mode used and added all the *Init* methods so one can reinitialize a given interface reference if necessary. This required to add *DECCipherBase* unit to the interface unit's uses clause.
- Fixed some regressions in *TFormat_HEX.Decode* and *TFormat_HEXL.Decode* methods for RawByteString.
- Added the SHA3 hash algorithms for Byte sized data and for messages not ending on complete bytes including Eric Grange's optimized assembler versions for Win32 and Win64.
- Fixed encoding and decoding errors in the *TCipher_SCOP* cipher. Somebody detected them during tests on CPUs with different byte order, conducted with FPC.
- Fixed a reported error in the initialization of *TCipher_Shark* as this might have failed depending on stack initialization.
- Added three cipher classes to provide compatibility with old and faulty versions of the XTEA, Shark and SCOP ciphers. The classes are named *TCipher_XTEA_DEC52*, *TCipher_Shark_DEC52* and *TCipher_SCOP_DEC52*. They are not registered in the class registration mechanism by default and due to their changed name, they have other identification values! If you want to register those you can use the *OLD_REGISTER_FAULTY_CIPHERS* define in *DECOptions.inc*.
- Fixed further use of syntax not supported by older Delphi versions

- Added *GetMinRounds* and *GetMaxRounds* properties to the following Hash algorithms: *THash_Tiger* (and its alias *THash_Tiger192*), *THash_HAVAL128*, *THash_HAVAL160*, *THash_HAVAL192*, *THash_HAVAL224*, *THash_HAVAL256*, *THash_Snefru128* and *THash_Snefru256*.
- Added an FMX based hash benchmark demo application.

7 Overview of changes made between DEC 6.1 and 6.1.1

.

Fixed usages of syntax not supported by Delphi versions older than XE7 thus preventing compilation with those older versions.

8 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.

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 into this new class. Made all hash implementations inherit from this one. Also moved the *IsPasswordHash* class method 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.

9 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 available but 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 will enforce these values of course, 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 or starting with DEC V6.2 as *TCipher_XTEA_DEC52.*

- 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.