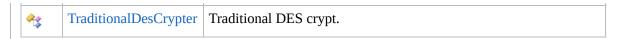
#### **CryptSharp Namespace**

CryptSharp provides a number of password crypt algorithms - BCrypt, LDAP, MD5 (and Apache's htpasswd variant), PHPass (WordPress, phpBB, Drupal), SHA256, SHA512, and Traditional and Extended DES. Additionally it includes Blowfish, SCrypt, and PBKDF2 for any HMAC (.NET's built-in PBKDF2 implementation supports only SHA-1).

If you are looking to store passwords, odds are, CryptSharp has the algorithms you need.

#### **Classes**

	Class	Description
<b>4</b> \$	BlowfishCrypter	Blowfish crypt, sometimes called BCrypt. A very good choice.
<b>*</b> F	Crypter	Crypts and verifies passwords. The main class for most uses of this library.
<del>\</del> \$	CrypterEnvironment	Lets you customize the list of crypt algorithms your program will accept.
<b>4</b> \$	CrypterOption	Options that modify the crypt operation.
<b>4</b> \$	CrypterOptionKey	The key type for options.
<b>4</b> \$	CrypterOptions	Stores options for the crypt operation.
<b>4</b> \$	CrypterProperty	Properties inherent to particular crypt algorithms.
<b>4</b> \$	ExtendedDesCrypter	Extended DES crypt.
<b>4</b> \$	LdapCrypter	LDAP schemes such as {SHA}.
<b>4</b> \$	LdapCrypterOption	Options that modify the LDAP crypt operation.
<b>P</b> \$	MD5Crypter	MD5 crypt, supported by nearly all systems. A variant supports Apache htpasswd files.
<b>4</b> 3	PhpassCrypter	PHPass crypt. Used by WordPress. Variants support phpBB and Drupal 7+.
<b>4</b> 3	Sha256Crypter	SHA256 crypt. A reasonable choice if you cannot use Blowfish crypt for policy reasons.
<b>4</b>	Sha512Crypter	SHA512 crypt. A reasonable choice if you cannot use Blowfish crypt for policy reasons.
43	ShaCrypter	Base class for Sha256Crypter and Sha512Crypter.



### **Enumerations**

	Enumeration	Description
yourself with Blowfish crypt variations if you have passwo generated pre-2011 using the C-language crypt_blowfish li a port thereof. CryptSharp was implemented from specifications.		Variations of the Blowfish crypt algorithm. You only need concern yourself with Blowfish crypt variations if you have passwords generated pre-2011 using the C-language crypt_blowfish library or a port thereof. CryptSharp was implemented from specification and is not a port, and therefore never had the bug these variants pertain to.
	LdapCrypterVariant	LDAP password schemes.
	MD5CrypterVariant	Modified versions of the MD5 crypt algorithm.
==	PhpassCrypterVariant	Modified versions of the PHPass crypt algorithm.

#### **BlowfishCrypter Class**

CryptSharp

Blowfish crypt, sometimes called BCrypt. A very good choice.

#### **Inheritance Hierarchy**

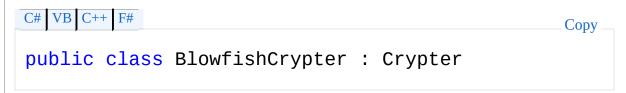
SystemObject CryptSharpCrypter

Crypt Sharp Blow fish Crypter

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

#### **Syntax**



The **BlowfishCrypter** type exposes the following members.

#### **Constructors**

	Name	Description
<b>≡</b>	BlowfishCrypter	Initializes a new instance of the <b>BlowfishCrypter</b> class

Top

#### **Methods**

	Name	Description
=0	CanCrypt	Checks if the particular crypt algorithm is compatible with the salt string or crypted password.  (Overrides CrypterCanCrypt(String).)
=0	Crypt(Byte)	Creates a one-way password hash (crypted password) from password bytes. (Inherited from Crypter.)
<b>≡</b>	Crypt(String)	Creates a one-way password hash (crypted password) from a password string. (Inherited from Crypter.)
<b>≅</b>	Crypt(Byte, String)	Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted

		passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords.  (Overrides CrypterCrypt(Byte, String).)
<b>≡</b>	Crypt(Byte, CrypterOptions)	Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation. (Inherited from Crypter.)
<b>≡</b>	Crypt(String, CrypterOptions)	Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)
≡₩	Crypt(String, String)	Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords.  (Inherited from Crypter.)
<b>≡</b>	GenerateSalt	Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)
<b>≓</b>	GenerateSalt(Int32)	Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password.  (Inherited from Crypter.)
<b>≡</b> ℚ	GenerateSalt(CrypterOptions)	Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form algorithm+salt+hash, so the salt is always carried along with the crypted password).

(Overrides CrypterGenerateSalt(CrypterOptions).)

#### Top

# **Properties**

Name	Description
	Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys. (Overrides CrypterProperties.)

#### Top

### **See Also**

#### Reference

CryptSharp Namespace

### **BlowfishCrypter Constructor**

CryptSharp

Initializes a new instance of the BlowfishCrypter class

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

#### **Syntax**

```
C# VB C++ F#

Copy

public BlowfishCrypter()
```

#### See Also

Reference

BlowfishCrypter Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# **BlowfishCrypter Methods**

The BlowfishCrypter type exposes the following members.

### Methods

	Name	Description
<b>≡</b>	CanCrypt	Checks if the particular crypt algorithm is compatible with the salt string or crypted password. (Overrides CrypterCanCrypt(String).)
<b>≡</b>	Crypt(Byte)	Creates a one-way password hash (crypted password) from password bytes.  (Inherited from Crypter.)
<b>≡</b>	Crypt(String)	Creates a one-way password hash (crypted password) from a password string.  (Inherited from Crypter.)
∃ 🍑	Crypt(Byte, String)	Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Overrides CrypterCrypt(Byte, String).)
<b>=</b>	Crypt(Byte, CrypterOptions)	Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation. (Inherited from Crypter.)
<b>=</b>	Crypt(String, CrypterOptions)	Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)
=	Crypt(String, String)	Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Inherited from Crypter.)

=	=0	GenerateSalt	Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)
-	•	GenerateSalt(Int32)	Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password.  (Inherited from Crypter.)
=	•	GenerateSalt(CrypterOptions)	Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form algorithm+salt+hash, so the salt is always carried along with the crypted password). (Overrides CrypterGenerateSalt(CrypterOptions).)

Top

#### **See Also**

Reference

BlowfishCrypter Class CryptSharp Namespace

#### **BlowfishCrypterCanCrypt Method**

CryptSharp

Checks if the particular crypt algorithm is compatible with the salt string or crypted password.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

#### **Syntax**

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com

CryptSharp Namespace

### **Overload List**

	Name	Description
<b>≡</b>	Crypt(Byte)	Creates a one-way password hash (crypted password) from password bytes. (Inherited from Crypter.)
=•	Crypt(String)	Creates a one-way password hash (crypted password) from a password string. (Inherited from Crypter.)
Ξ₩	Crypt(Byte, String)	Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using <a href="GenerateSalt(CrypterOptions">GenerateSalt(CrypterOptions</a> ). Because crypted passwords take the form <a href="algorithm+salt+hash">algorithm+salt+hash</a> , if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Overrides CrypterCrypt(Byte, String).)
<b>≡</b>	Crypt(Byte, CrypterOptions)	Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation. (Inherited from Crypter.)
<b>≡</b>	Crypt(String, CrypterOptions)	Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)
₫•	Crypt(String, String)	Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Inherited from Crypter.)

Top

#### See Also

#### Reference

BlowfishCrypter Class CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# BlowfishCrypterCrypt Method (Byte, String)

CryptSharp

Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using

GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate \*and\* verify crypted passwords.

Namespace: CryptSharp

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

#### **Syntax**

```
C# VB C++ F#
                                                        Copy
public override string Crypt(
          byte[] password,
          string salt
 )
Parameters
```

password

Type: SystemByte

The bytes of the password.

salt

Type: SystemString

The salt string or crypted password containing a salt string.

#### **Return Value**

Type: String

The crypted password.

#### See Also

#### Reference

BlowfishCrypter Class

**Crypt Overload** 

# CryptSharp Namespace

#### **Overload List**

	Name	Description
=0	GenerateSalt	Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)
₩	GenerateSalt(Int32)	Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password.  (Inherited from Crypter.)
≅ 🍑	GenerateSalt(CrypterOptions)	Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form algorithm+salt+hash, so the salt is always carried along with the crypted password). (Overrides CrypterGenerateSalt(CrypterOptions).)

Top

#### See Also

Reference

BlowfishCrypter Class CryptSharp Namespace

# BlowfishCrypterGenerateSalt Method (CrypterOptions)

CryptSharp

Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form algorithm+salt+hash, so the salt is always carried along with the crypted password).

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

#### **Syntax**

```
C# VB C++ F#
                                                      Copy
public override string GenerateSalt(
          CrypterOptions options
Parameters
```

options

Type: CryptSharpCrypterOptions

Options modifying the salt generation.

#### **Return Value**

Type: String The salt string.

#### See Also

#### Reference

BlowfishCrypter Class

GenerateSalt Overload

CryptSharp Namespace

# **BlowfishCrypter Properties**

CryptSharp

The BlowfishCrypter type exposes the following members.

### **Properties**

Name Description		Description
		Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys. (Overrides CrypterProperties.)

Top

#### See Also

Reference

BlowfishCrypter Class CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

#### **BlowfishCrypterProperties Property**

CryptSharp

Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

#### **Syntax**



#### **BlowfishCrypterVariant Enumeration**

CryptSharp

Variations of the Blowfish crypt algorithm. You only need concern yourself with Blowfish crypt variations if you have passwords generated pre-2011 using the C-language crypt\_blowfish library or a port thereof. CryptSharp was implemented from specification and is not a port, and therefore never had the bug these variants pertain to.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

#### **Syntax**



Copy

public enum BlowfishCrypterVariant

#### **Members**

	Member name	Value	Description
	Unspecified	0	The \$2a\$ prefix indicates nothing about whether or not the crypted password was created with a pre-2011 version of the C-language crypt_blowfish library. Pre-2011, that library had a sign extension bug affecting non-ASCII passwords. See <a href="mailto:EmulateCryptBlowfishSignExtensionBug">EmulateCryptBlowfishSignExtensionBug</a> for a more detailed explanation of the bug in question.
	Compatible	1	The \$2x\$ prefix indicates that these passwords were generated with pre-2011 crypt_blowfish or a port originating from it. If you have old crypted non-ASCII passwords you can't re-derive, and still want to verify them with CryptSharp, ensure that they have the \$2x\$ prefix instead of the \$2a\$ prefix. This will indicate to CryptSharp that it should emulate the bug when verifying the password.
	Corrected	2	The \$2y\$ prefix indicates that pre-2011 crypt_blowfish's sign extension bug does not affect these crypted passwords. For passwords crypted with CryptSharp, this has always been true and as such selecting this variant changes the prefix but otherwise does not affect the output.

#### See Also

Reference

CryptSharp Namespace

Crypts and verifies passwords. The main class for most uses of this library.

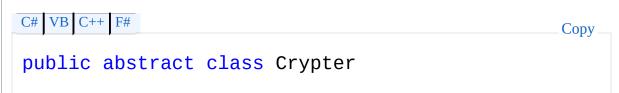
#### **Inheritance Hierarchy**

SystemObject CryptSharpCrypter More...

Namespace: CryptSharp

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

#### **Syntax**



The **Crypter** type exposes the following members.

#### **Methods**

	Name	Description
<b>≅</b>	CanCrypt	Checks if the particular crypt algorithm is compatible with the salt string or crypted password.
ĕ∳S	CheckPassword(Byte, String)	Checks if the crypted password matches the given password bytes.
ĕ∳S	CheckPassword(String, String)	Checks if the crypted password matches the given password string.
<b>≅</b>	Crypt(Byte)	Creates a one-way password hash (crypted password) from password bytes.
<b>≡</b>	Crypt(String)	Creates a one-way password hash (crypted password) from a password string.
<b>≡</b>	Crypt(Byte, CrypterOptions)	Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation.
₩	Crypt(Byte, String)	Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password.

		Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords.
<b>≡</b>	Crypt(String, CrypterOptions)	Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation.
₫•••	Crypt(String, String)	Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords.
=	GenerateSalt	Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password.
=0	GenerateSalt(Int32)	Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password.
=••	GenerateSalt(CrypterOptions)	Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form algorithm+salt+hash, so the salt is always carried along with the crypted password).
=₩	<b>GetCrypter</b>	Searches for a crypt algorithm compatible with the specified crypted password or prefix.

# Top

# **Properties**

	Name	Description
s s	Blowfish	Blowfish crypt, sometimes called BCrypt. A very good choice.

<b>≝</b> S	ExtendedDes	Extended DES crypt.	
≝°s	Ldap	LDAP schemes such as {SHA}.	
<b>₹</b> 5	MD5	MD5 crypt, supported by nearly all systems. A variant supports Apache htpasswd files.	
≝°s	Phpass	PHPass crypt. Used by WordPress. Variants support phpBB and Drupal 7+.	
	Properties	Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys.	
<b>₹</b>	Sha256	SHA256 crypt. A reasonable choice if you cannot use Blowfish crypt for policy reasons.	
<b>₹</b>	Sha512	SHA512 crypt. A reasonable choice if you cannot use Blowfish crypt for policy reasons.	
<b>≅</b> S	TraditionalDes	Traditional DES crypt.	

Top

#### **Examples**

```
Crypting a Password
                                                   Copy
using CryptSharp;
// Crypt using the Blowfish crypt ("BCrypt") algorit
string cryptedPassword = Crypter.Blowfish.Crypt(pass
Checking a Password
                                                   Copy
using CryptSharp;
// Do the passwords match?
// You can also check a password using the Crypt met
bool matches = Crypter.CheckPassword(testPassword, c
Specifying Options
                                                   Copy
using CryptSharp;
// Specify the $apr1$ Apache htpasswd variant of the
string cryptedPassword = Crypter.MD5.Crypt(password,
```

```
{
      { CrypterOption.Variant, MD5CrypterVariant.A
});
```

#### See Also

Reference

CryptSharp Namespace

#### **Inheritance Hierarchy**

SystemObject

#### CryptSharpCrypter

Crypt Sharp Blow fish Crypter

Crypt Sharp Extended Des Crypter

CryptSharpLdapCrypter

CryptSharpMD5Crypter

CryptSharpPhpassCrypter

CryptSharpShaCrypter

Crypt Sharp Traditional Des Crypter

Copyright 2010-2014 James F. Bellinger

# **Crypter Methods**

The Crypter type exposes the following members.

# Methods

	Name	Description
<b>≡</b>	CanCrypt	Checks if the particular crypt algorithm is compatible with the salt string or crypted password.
<b>=\$S</b>	CheckPassword(Byte, String)	Checks if the crypted password matches the given password bytes.
<b>=\$</b> S	CheckPassword(String, String)	Checks if the crypted password matches the given password string.
<b>≡</b>	Crypt(Byte)	Creates a one-way password hash (crypted password) from password bytes.
<b>≡</b>	Crypt(String)	Creates a one-way password hash (crypted password) from a password string.
<b>≡</b>	Crypt(Byte, CrypterOptions)	Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation.
<b>≓</b>	Crypt(Byte, String)	Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords.
<b>≡</b>	Crypt(String, CrypterOptions)	Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation.
ΞΦ	Crypt(String, String)	Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords.
<b>=</b>	GenerateSalt	Generates a salt string with default options. The purpose of

		salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password.
<b>≟</b>	GenerateSalt(Int32)	Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password.
- II 🍑	GenerateSalt(CrypterOptions)	Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form algorithm+salt+hash, so the salt is always carried along with the crypted password).
≡ŵS	GetCrypter	Searches for a crypt algorithm compatible with the specified crypted password or prefix.

Top

### See Also

Reference

**Crypter Class** 

CryptSharp Namespace

#### **CrypterCanCrypt Method**

CryptSharp

Checks if the particular crypt algorithm is compatible with the salt string or crypted password.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

#### **Syntax**

```
public abstract bool CanCrypt(
    string salt
)

Parameters
salt
    Type: SystemString
    The salt string or crypted password.

Return Value
Type: Boolean
true if the algorithm is compatible.

See Also
Reference
Crypter Class
```

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com

CryptSharp Namespace

# **CrypterCheckPassword Method**

CryptSharp

#### **Overload List**

	Name	Description
<b>=\$S</b>	CheckPassword(Byte, String)	Checks if the crypted password matches the given password bytes.
<b>=\$S</b>	CheckPassword(String, String)	Checks if the crypted password matches the given password string.

Top

#### **See Also**

Reference

**Crypter Class** 

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# **CrypterCheckPassword Method (Byte, String)**

CryptSharp

Checks if the crypted password matches the given password bytes.

Namespace: CryptSharp

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

#### **Syntax**

```
C# VB C++ F#
                                                              Copy
 public static bool CheckPassword(
           byte[] password,
           string cryptedPassword
 )
Parameters
password
  Type: SystemByte
  The password bytes to test.
cryptedPassword
  Type: SystemString
  The crypted password.
Return Value
Type: Boolean
true if the passwords match.
See Also
Reference
Crypter Class
CheckPassword Overload
CryptSharp Namespace
```

Copyright 2010-2014 James F. Bellinger

# **CrypterCheckPassword Method (String, String)**

CryptSharp

Checks if the crypted password matches the given password string.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

#### **Syntax**

```
C# VB C++ F#
                                                               Copy
 public static bool CheckPassword(
           string password,
           string cryptedPassword
 )
Parameters
password
  Type: SystemString
  The password string to test. Characters are UTF-8 encoded.
cryptedPassword
  Type: SystemString
  The crypted password.
Return Value
Type: Boolean
true if the passwords match.
See Also
Reference
Crypter Class
CheckPassword Overload
```

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com

CryptSharp Namespace

#### **Overload List**

	Name	Description
<b>≡</b>	Crypt(Byte)	Creates a one-way password hash (crypted password) from password bytes.
<b>≡</b>	Crypt(String)	Creates a one-way password hash (crypted password) from a password string.
=•	Crypt(Byte, CrypterOptions)	Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation.
€₩	Crypt(Byte, String)	Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords.
<b>≡</b>	Crypt(String, CrypterOptions)	Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation.
≅ 🕡	Crypt(String, String)	Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords.

Top

#### See Also

Reference

**Crypter Class** 

CryptSharp Namespace

#### **CrypterCrypt Method (Byte)**

CryptSharp

Creates a one-way password hash (crypted password) from password bytes.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

```
Syntax
```

```
public string Crypt(
          byte[] password
)
```

#### **Parameters**

password

Type: SystemByte

The bytes of the password.

#### **Return Value**

Type: String

The crypted password.

#### **See Also**

Reference

**Crypter Class** 

**Crypt Overload** 

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

#### **CrypterCrypt Method (String)**

CryptSharp

Creates a one-way password hash (crypted password) from a password string.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

#### **Syntax**

#### **Parameters**

password

Type: SystemString

The password string. Characters are UTF-8 encoded.

#### **Return Value**

Type: String

The crypted password.

#### **See Also**

Reference

**Crypter Class** 

**Crypt Overload** 

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# CrypterCrypt Method (Byte, CrypterOptions)

CryptSharp

Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

```
Syntax
```

```
C# VB C++ F#
                                                                Copy
 public string Crypt(
            byte[] password,
           CrypterOptions options
 )
Parameters
password
  Type: SystemByte
  The bytes of the password.
options
  Type: CryptSharpCrypterOptions
  Options modifying the crypt operation.
Return Value
Type: String
The crypted password.
See Also
Reference
Crypter Class
```

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com

**Crypt Overload** 

CryptSharp Namespace

Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using

GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate \*and\* verify crypted passwords.

Namespace: CryptSharp

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

#### **Syntax**

#### Return Value

Type: String

The crypted password.

#### **See Also**

#### Reference

**Crypter Class** 

**Crypt Overload** 

# CryptSharp Namespace

# **CrypterCrypt Method (String, CrypterOptions**)

CryptSharp

Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**

```
C# VB C++ F#
                                                        Copy
 public string Crypt(
          string password,
          CrypterOptions options
 )
Parameters
password
```

Type: SystemString

The password string. Characters are UTF-8 encoded.

options

Type: CryptSharpCrypterOptions

Options modifying the crypt operation.

## **Return Value**

Type: String

The crypted password.

# See Also

#### Reference

Crypter Class

**Crypt Overload** 

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using

GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate \*and\* verify crypted passwords.

Namespace: CryptSharp

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**

```
C# VB C++ F#
                                                                  Copy
 public string Crypt(
            string password,
            string salt
 )
Parameters
password
  Type: SystemString
  The password string. Characters are UTF-8 encoded.
salt
  Type: SystemString
  The salt string or crypted password containing a salt string.
Return Value
Type: String
The crypted password.
```

## See Also

Reference

**Crypter Class** 

**Crypt Overload** 

# CryptSharp Namespace

# **Overload List**

	Name	Description
<b>≅</b>	GenerateSalt	Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password.
=₩	GenerateSalt(Int32)	Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password.
≅ 🗳	GenerateSalt(CrypterOptions)	Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form algorithm+salt+hash, so the salt is always carried along with the crypted password).

Top

# **See Also**

Reference

**Crypter Class** 

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# **CrypterGenerateSalt Method**

CryptSharp

Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password.

Namespace: CryptSharp

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**



Copyright 2010-2014 James F. Bellinger

# **CrypterGenerateSalt Method (Int32)**

CryptSharp

Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**

```
C# VB C++ F#
                                                                Copy
 public string GenerateSalt(
            int rounds
 )
Parameters
rounds
  Type: SystemInt32
Return Value
Type: String
The salt string.
See Also
Reference
Crypter Class
GenerateSalt Overload
CryptSharp Namespace
```

Copyright 2010-2014 James F. Bellinger

# CrypterGenerateSalt Method (CrypterOptions)

CryptSharp

Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form algorithm+salt+hash, so the salt is always carried along with the crypted password).

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**

## **Parameters**

options

Type: CryptSharpCrypterOptions

Options modifying the salt generation.

## **Return Value**

Type: String
The salt string.

## See Also

#### Reference

**Crypter Class** 

GenerateSalt Overload

CryptSharp Namespace

# **CrypterGetCrypter Method**

CryptSharp

Searches for a crypt algorithm compatible with the specified crypted password or prefix.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**

## **Parameters**

cryptedPassword

Type: SystemString

The crypted password or prefix.

## **Return Value**

Type: Crypter

A compatible crypt algorithm.

# **Exceptions**

Exception	Condition
ArgumentException	No compatible crypt algorithm was found.

# **See Also**

## Reference

**Crypter Class** 

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

The Crypter type exposes the following members.

# **Properties**

	Name	Description
s s	Blowfish	Blowfish crypt, sometimes called BCrypt. A very good choice.
s s	ExtendedDes	Extended DES crypt.
s s	Ldap	LDAP schemes such as {SHA}.
<b>₽</b> S	MD5	MD5 crypt, supported by nearly all systems. A variant supports Apache htpasswd files.
s s	Phpass	PHPass crypt. Used by WordPress. Variants support phpBB and Drupal 7+.
	Properties	Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys.
<b>₽</b> S	Sha256	SHA256 crypt. A reasonable choice if you cannot use Blowfish crypt for policy reasons.
<b>₽</b> S	Sha512	SHA512 crypt. A reasonable choice if you cannot use Blowfish crypt for policy reasons.
s s	TraditionalDes	Traditional DES crypt.

Top

# **See Also**

Reference

**Crypter Class** 

CryptSharp Namespace

# **CrypterBlowfish Property**

CryptSharp

Blowfish crypt, sometimes called BCrypt. A very good choice.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**

```
C# VB C++ F#

public static BlowfishCrypter Blowfish { get; }

Property Value
Type: BlowfishCrypter

See Also

Reference
Crypter Class
CryptSharp Namespace
```

Copyright 2010-2014 James F. Bellinger

# **CrypterExtendedDes Property**

CryptSharp

Extended DES crypt.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**

```
C# VB C++ F#

public static ExtendedDesCrypter ExtendedDes { get;

Property Value
Type: ExtendedDesCrypter
```

# **See Also**

Reference

**Crypter Class** 

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# **CrypterLdap Property**

CryptSharp

LDAP schemes such as {SHA}.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**

```
C# VB C++ F#

public static LdapCrypter Ldap { get; }

Property Value
Type: LdapCrypter

See Also

Reference
Crypter Class
CryptSharp Namespace
```

Copyright 2010-2014 James F. Bellinger

# **CrypterMD5 Property**

CryptSharp

MD5 crypt, supported by nearly all systems. A variant supports Apache htpasswd files.

Namespace: CryptSharp

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**

```
public static MD5Crypter MD5 { get; }

Property Value
Type: MD5Crypter

See Also
Reference
Crypter Class
CryptSharp Namespace
```

# **CrypterPhpass Property**

CryptSharp

Copy

PHPass crypt. Used by WordPress. Variants support phpBB and Drupal 7+.

Namespace: CryptSharp

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# Syntax C# VB C++ F# public static PhpassCrypter Phpass { get; } Property Value

Type: PhpassCrypter

**See Also** 

Reference

**Crypter Class** 

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# **CrypterProperties Property**

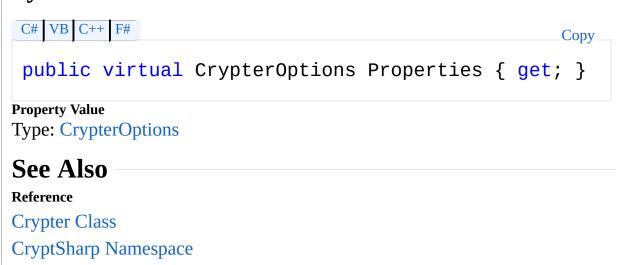
CryptSharp

Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys.

Namespace: CryptSharp

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**



# **CrypterSha256 Property**

CryptSharp

SHA256 crypt. A reasonable choice if you cannot use Blowfish crypt for policy reasons.

Namespace: CryptSharp

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**

```
public static Sha256Crypter Sha256 { get; }

Property Value
Type: Sha256Crypter

See Also
Reference
Crypter Class
CryptSharp Namespace
```

# **CrypterSha512 Property**

CryptSharp

SHA512 crypt. A reasonable choice if you cannot use Blowfish crypt for policy reasons.

Namespace: CryptSharp

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**

```
C# VB C++ F#

public static Sha512Crypter Sha512 { get; }

Property Value
Type: Sha512Crypter

See Also

Reference
Crypter Class
CryptSharp Namespace
```

# **CrypterTraditionalDes Property**

CryptSharp

Traditional DES crypt.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**



See Also

Reference

**Crypter Class** 

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# **CrypterEnvironment Class**

CryptSharp

Lets you customize the list of crypt algorithms your program will accept.

# **Inheritance Hierarchy**

SystemObject CryptSharpCrypterEnvironment

Namespace: CryptSharp

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**

C# VB C++ F#

Copy

public class CrypterEnvironment

The **CrypterEnvironment** type exposes the following members.

# **Constructors**

	Name	Description
<b>≡</b>	CrypterEnvironment	Initializes a new instance of the <b>CrypterEnvironment</b> class

Top

## **Methods**

	Name	Description
<b>=</b>	CheckPassword(Byte, String)	Checks if the crypted password matches the given password bytes.
<b>=</b>	CheckPassword(String, String)	Checks if the crypted password matches the given password string.
<b>≡</b>	GetCrypter	Searches for a crypt algorithm compatible with the specified crypted password or prefix.
<b>≡</b>	MakeReadOnly	Prevents future changes to the environment.
<b>≡</b>	TryGetCrypter	Searches for a crypt algorithm compatible with the specified crypted password or prefix,

Top

# **Properties**

	Name	Description
	Crypters	The collection of crypters in this environment.
s s	Default	The default environment.
	IsReadOnly	true if the environment cannot be changed.

# Top

# **See Also**

Reference

CryptSharp Namespace

# **CrypterEnvironment Constructor**

CryptSharp

Initializes a new instance of the CrypterEnvironment class

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**

```
C# VB C++ F#

public CrypterEnvironment()
```

# See Also

Reference

CrypterEnvironment Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# **CrypterEnvironment Methods**

CryptSharp

The CrypterEnvironment type exposes the following members.

# **Methods**

	Name	Description
<b>≡</b>	CheckPassword(Byte, String)	Checks if the crypted password matches the given password bytes.
<b>≡</b>	CheckPassword(String, String)	Checks if the crypted password matches the given password string.
<b>=</b>	GetCrypter	Searches for a crypt algorithm compatible with the specified crypted password or prefix.
<b>≡</b>	MakeReadOnly	Prevents future changes to the environment.
<b>=</b>	TryGetCrypter	Searches for a crypt algorithm compatible with the specified crypted password or prefix,

Top

# **See Also**

Reference

CrypterEnvironment Class

CryptSharp Namespace

# **CrypterEnvironmentCheckPassword Method**

CryptSharp

# **Overload List**

	Name	Description
<b>≡</b>	CheckPassword(Byte, String)	Checks if the crypted password matches the given password bytes.
<b>=</b>	CheckPassword(String, String)	Checks if the crypted password matches the given password string.

Top

# See Also

Reference

CrypterEnvironment Class

CryptSharp Namespace

# CrypterEnvironmentCheckPassword Method (Byte, String)

CryptSharp

Checks if the crypted password matches the given password bytes.

Namespace: CryptSharp

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**

```
C# VB C++ F#
                                                              Copy
 public bool CheckPassword(
           byte[] password,
           string cryptedPassword
 )
Parameters
password
  Type: SystemByte
  The password bytes to test.
cryptedPassword
  Type: SystemString
  The crypted password.
Return Value
Type: Boolean
true if the passwords match.
See Also
Reference
CrypterEnvironment Class
CheckPassword Overload
CryptSharp Namespace
```

Copyright 2010-2014 James F. Bellinger

# CrypterEnvironmentCheckPassword Method (String, String)

CryptSharp

Checks if the crypted password matches the given password string.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**

```
C# VB C++ F#
                                                               Copy
 public bool CheckPassword(
           string password,
           string cryptedPassword
 )
Parameters
password
  Type: SystemString
  The password string to test. Characters are UTF-8 encoded.
cryptedPassword
  Type: SystemString
  The crypted password.
Return Value
Type: Boolean
true if the passwords match.
See Also
Reference
CrypterEnvironment Class
CheckPassword Overload
CryptSharp Namespace
```

Copyright 2010-2014 James F. Bellinger

# CrypterEnvironmentGetCrypter Method CryptSharp

Searches for a crypt algorithm compatible with the specified crypted password or prefix.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**

#### **Parameters**

cryptedPassword

Type: SystemString

The crypted password or prefix.

## **Return Value**

Type: Crypter

A compatible crypt algorithm.

# **Exceptions**

Exception	Condition
ArgumentException	No compatible crypt algorithm was found.

# **See Also**

## Reference

CrypterEnvironment Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# **CrypterEnvironmentMakeReadOnly Method**

CryptSharp

Prevents future changes to the environment.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**



## **Return Value**

Type: CrypterEnvironment

The same CrypterEnvironment.

# See Also

## Reference

CrypterEnvironment Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# CrypterEnvironmentTryGetCrypter Method

CryptSharp

Searches for a crypt algorithm compatible with the specified crypted password or prefix,

Namespace: CryptSharp

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**

```
C# VB C++ F#
                                                      Copy
public bool TryGetCrypter(
          string cryptedPassword,
          out Crypter crypter
 )
Parameters
```

cryptedPassword

Type: SystemString

The crypted password or prefix.

crypter

Type: CryptSharpCrypter

A compatible crypt algorithm.

## **Return Value**

Type: Boolean

true if a compatible crypt algorithm was found.

# See Also

#### Reference

CrypterEnvironment Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# **CrypterEnvironment Properties**

CryptSharp

The CrypterEnvironment type exposes the following members.

# **Properties**

	Name	Description
	Crypters	The collection of crypters in this environment.
<b>≝</b> s	Default	The default environment.
	IsReadOnly	true if the environment cannot be changed.

Top

# **See Also**

Reference

CrypterEnvironment Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# **CrypterEnvironmentCrypters Property**

CryptSharp

The collection of crypters in this environment.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**

```
C# VB C++ F#

public IList<Crypter> Crypters { get; }

Property Value
Type: IListCrypter

See Also
Reference
CrypterEnvironment Class
```

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com

CryptSharp Namespace

# **CrypterEnvironmentDefault Property**

CryptSharp

The default environment.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**

```
C# VB C++ F#

Copy

public static CrypterEnvironment Default { get; }

Property Value
```

Type: CrypterEnvironment

# **See Also**

Reference

CrypterEnvironment Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# CrypterEnvironmentIsReadOnly Property CryptSharp

true if the environment cannot be changed.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**

```
public bool IsReadOnly { get; }

Property Value
Type: Boolean

See Also

Reference
CrypterEnvironment Class
CryptSharp Namespace
```

Copyright 2010-2014 James F. Bellinger

# **CrypterOption Class**

CryptSharp

Options that modify the crypt operation.

# **Inheritance Hierarchy**

 ${\bf SystemObject~CryptSharpCrypterOption}$ 

CryptSharpLdapCrypterOption

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**



The **CrypterOption** type exposes the following members.

# **Fields**

	Name	Description
₽ S	Rounds	The number of rounds to iterate.
₽ S	Variant	The variant of the crypt algorithm to use.

Top

## **See Also**

Reference

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# **CrypterOption Fields**

CryptSharp

The CrypterOption type exposes the following members.

# **Fields**

	Name	Description
₽S	Rounds	The number of rounds to iterate.
<b>♦ S</b>	Variant	The variant of the crypt algorithm to use.

## Top

# See Also

## Reference

CrypterOption Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# **CrypterOptionRounds Field**

CryptSharp

The number of rounds to iterate.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**

C# VB C++ F#

public static readonly CrypterOptionKey Rounds

Field Value

Type: CrypterOptionKey

# See Also

Reference

CrypterOption Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# **CrypterOptionVariant Field**

CryptSharp

The variant of the crypt algorithm to use.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

#### **Syntax**

C# VB C++ F#

Copy

public static readonly CrypterOptionKey Variant

Field Value

Type: CrypterOptionKey

#### **See Also**

Reference

CrypterOption Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# **CrypterOptionKey Class**

CryptSharp

The key type for options.

#### **Inheritance Hierarchy**

SystemObject CryptSharpCrypterOptionKey

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## **Syntax**

```
C# VB C++ F# Copy

public class CrypterOptionKey
```

The **CrypterOptionKey** type exposes the following members.

#### **Constructors**

	Name	Description
<b>≡</b>	CrypterOptionKey	Creates a new option key.

Top

#### **Methods**

	Name	Description
<b>≡</b>	CheckValue	Throws an exception if the value is incompatible with this option.
<b>≡</b>	ToString	Returns a String that represents the current Object. (Overrides ObjectToString.)

Top

#### **Properties**

	Name	Description
•	Description	A description of the option.
1	ValueType	The type of the option's value.

#### Top

# See Also

Reference

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger Send comments on this topic to jfb@zer7.com

## **CrypterOptionKey Constructor**

CryptSharp

Creates a new option key.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## **Syntax**

#### **Parameters**

description

Type: SystemString

A description of the option.

valueType

Type: SystemType

The type of the option's value.

#### **See Also**

Reference

CrypterOptionKey Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# **CrypterOptionKey Methods**

CryptSharp

The CrypterOptionKey type exposes the following members.

# **Methods**

	Name	Description	
<b>≡</b>	CheckValue	Throws an exception if the value is incompatible with this option.	
<b>≡</b>	ToString	Returns a String that represents the current Object. (Overrides ObjectToString.)	

#### Top

# See Also

#### Reference

CrypterOptionKey Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# CrypterOptionKeyCheckValue Method

CryptSharp

Throws an exception if the value is incompatible with this option.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## **Syntax**

```
C# VB C++ F#
                                                         Copy
 public void CheckValue(
          Object value
Parameters
value
```

Type: SystemObject The value to check.

#### See Also

Reference

CrypterOptionKey Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# CrypterOptionKeyToString Method

CryptSharp

Returns a String that represents the current Object.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## **Syntax**

```
C# VB C++ F#

Copy

public override string ToString()
```

**Return Value** 

Type: String

A String that represents the current Object.

#### **See Also**

Reference

CrypterOptionKey Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# **CrypterOptionKey Properties**

CryptSharp

The CrypterOptionKey type exposes the following members.

# **Properties**

	Name	Description
	Description	A description of the option.
	ValueType	The type of the option's value.

#### Top

#### **See Also**

#### Reference

CrypterOptionKey Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# CrypterOptionKeyDescription Property

CryptSharp

A description of the option.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## **Syntax**

```
C# VB C++ F#

Copy

public string Description { get; }

Property Value
Type: String

See Also

Reference

CrypterOptionKey Class

CryptSharp Namespace
```

Copyright 2010-2014 James F. Bellinger

# **CrypterOptionKeyValueType Property**

CryptSharp

The type of the option's value.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## **Syntax**

```
C# VB C++ F#

Copy

public Type ValueType { get; }

Property Value
Type: Type

See Also

Reference
CrypterOptionKey Class
CryptSharp Namespace
```

Copyright 2010-2014 James F. Bellinger

# **CrypterOptions Class**

CryptSharp

Stores options for the crypt operation.

#### **Inheritance Hierarchy**

SystemObject CryptSharpCrypterOptions

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

#### **Syntax**

The **CrypterOptions** type exposes the following members.

#### **Constructors**

	Name	Description
<b>≡</b>	CrypterOptions	Initializes a new instance of the <b>CrypterOptions</b> class

Top

#### **Methods**

	Name	Description
<b>≡</b>	Add	Sets the value of an option, if the option has not already been set.
<b>≡</b>	Clear	Clears all options.
<b>≡</b>	ContainsKey	Checks if an option is set.
<b>≡</b>	GetEnumerator	Returns an enumerator that iterates through all options.
<b>≡</b>	GetValueT(CrypterOptionKey)	Gets the value of an option, if the option is set, or a default value otherwise.
<b>≡</b>	GetValueT(CrypterOptionKey, T)	Gets the value of an option, if the option is set, or a specified default value otherwise.
<b>≡</b>	MakeReadOnly	Prevents future changes to the options.

<b>≡</b>	Remove	Clears an option.
<b>≡</b>	TryGetValue	Gets the value of an option, if the option is set.

#### Top

# **Properties**

	Name	Description
	Count	The number of options that have been set.
	IsReadOnly	true if the options cannot be changed.
	Item	Gets or sets an option.
<b>≧</b> S	None	No options.

#### Top

# **See Also**

Reference

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger Send comments on this topic to jfb@zer7.com

# **CrypterOptions Constructor**

CryptSharp

Initializes a new instance of the CrypterOptions class

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## **Syntax**

```
C# VB C++ F#

Copy

public CrypterOptions()
```

#### See Also

Reference

CrypterOptions Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

The CrypterOptions type exposes the following members.

## **Methods**

	Name	Description
<b>=</b>	Add	Sets the value of an option, if the option has not already been set.
<b>≡</b>	Clear	Clears all options.
<b>≡</b>	ContainsKey	Checks if an option is set.
<b>≡</b>	GetEnumerator	Returns an enumerator that iterates through all options.
<b>≡</b>	GetValueT(CrypterOptionKey)	Gets the value of an option, if the option is set, or a default value otherwise.
<b>≡</b>	GetValueT(CrypterOptionKey, T)	Gets the value of an option, if the option is set, or a specified default value otherwise.
<b>≡</b>	MakeReadOnly	Prevents future changes to the options.
<b>≡</b>	Remove	Clears an option.
<b>≡</b>	TryGetValue	Gets the value of an option, if the option is set.

#### Top

# **See Also**

Reference

**CrypterOptions Class** 

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# **CrypterOptionsAdd Method**

CryptSharp

Sets the value of an option, if the option has not already been set.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

```
Syntax
```

```
C# VB C++ F#
                                                    Copy
public void Add(
         CrypterOptionKey key,
         Object value
)
```

#### **Parameters**

kev

Type: CryptSharpCrypterOptionKey

The key of the option.

value

Type: SystemObject

The value of the option.

#### See Also

Reference

CrypterOptions Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# **CrypterOptionsClear Method**

CryptSharp

Clears all options.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## **Syntax**

```
C# VB C++ F#

Copy

public void Clear()
```

#### See Also

Reference

CrypterOptions Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

## **CrypterOptionsContainsKey Method**

CryptSharp

Checks if an option is set.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

#### **Syntax**

#### Parameters

key

Type: CryptSharpCrypterOptionKey

The key of the option.

#### **Return Value**

Type: Boolean

true if the option is set.

#### See Also

Reference

CrypterOptions Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# **CrypterOptionsGetEnumerator Method**

CryptSharp

Returns an enumerator that iterates through all options.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

#### **Syntax**

C# VB C++ F#

Copy

public IEnumerator<KeyValuePair<CrypterOptionKey, Ob</pre>

Return Value

Type: IEnumeratorKeyValuePairCrypterOptionKey, Object

An enumerator for the options.

**Implements** 

IEnumerableTGetEnumerator

#### See Also

Reference

CrypterOptions Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# **CrypterOptionsGetValue Method**

CryptSharp

# **Overload List**

	Name	Description
<b>=</b>	GetValueT(CrypterOptionKey)	Gets the value of an option, if the option is set, or a default value otherwise.
=•	GetValueT(CrypterOptionKey, T)	Gets the value of an option, if the option is set, or a specified default value otherwise.

Top

#### **See Also**

Reference

CrypterOptions Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# CrypterOptionsGetValueT Method (CrypterOptionKey)

CryptSharp

Gets the value of an option, if the option is set, or a default value otherwise.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

#### **Syntax**

#### **See Also**

#### Reference

CrypterOptions Class

GetValue Overload

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# CrypterOptionsGetValueT Method (CrypterOptionKey, T)

CryptSharp

Gets the value of an option, if the option is set, or a specified default value otherwise.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

```
Syntax
```

```
C# VB C++ F#
                                                                  Copy
 public T GetValue<T>(
            CrypterOptionKey key,
            T defaultValue
 )
Parameters
key
  Type: CryptSharpCrypterOptionKey
  The key of the option.
defaultValue
  Type: T
  The default value if the option is not set.
Type Parameters
  The type of the option's value.
Return Value
Type: T
The option's value.
See Also
Reference
CrypterOptions Class
GetValue Overload
CryptSharp Namespace
```

Copyright 2010-2014 James F. Bellinger

# CrypterOptionsMakeReadOnly Method

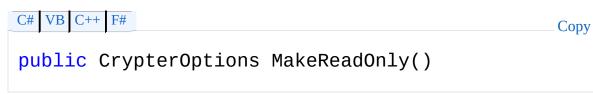
CryptSharp

Prevents future changes to the options.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## **Syntax**



#### **Return Value**

Type: CrypterOptions

The same CrypterOptions.

#### See Also

Reference

CrypterOptions Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# **CrypterOptionsRemove Method**

CryptSharp

Clears an option.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

#### **Syntax**

#### Parameters

key

Type: CryptSharpCrypterOptionKey

The key of the option.

#### **Return Value**

Type: Boolean

true if the option was found and cleared.

#### See Also

Reference

CrypterOptions Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

## **CrypterOptionsTryGetValue Method**

CryptSharp

Gets the value of an option, if the option is set.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

```
Syntax
```

#### **Parameters**

key

Type: CryptSharpCrypterOptionKey

The key of the option.

value

Type: SystemObject

The value, or null if the option is not set.

#### **Return Value**

Type: Boolean

true if the option is set.

#### See Also

#### Reference

CrypterOptions Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# **CrypterOptions Properties**

CryptSharp

The CrypterOptions type exposes the following members.

# **Properties**

	Name	Description
	Count	The number of options that have been set.
	IsReadOnly	true if the options cannot be changed.
	Item	Gets or sets an option.
≝°s	None	No options.

Top

#### **See Also**

Reference

**CrypterOptions Class** 

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# **CrypterOptionsCount Property**

CryptSharp

The number of options that have been set.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

```
Syntax
```

```
C# VB C++ F#

Copy

public int Count { get; }

Property Value
Type: Int32

See Also

Reference
CrypterOptions Class
CryptSharp Namespace
```

Copyright 2010-2014 James F. Bellinger

# **CrypterOptionsIsReadOnly Property**

CryptSharp

true if the options cannot be changed.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## **Syntax**

```
C# VB C++ F#

public bool IsReadOnly { get; }

Property Value
Type: Boolean

See Also

Reference
CrypterOptions Class
CryptSharp Namespace
```

Copyright 2010-2014 James F. Bellinger

## **CrypterOptionsItem Property**

CryptSharp

Gets or sets an option.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

#### **Syntax**

#### Parameters

key

Type: CryptSharpCrypterOptionKey

The key of the option.

#### **Return Value**

Type: Object

The value of the option.

#### See Also

Reference

CrypterOptions Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# **CrypterOptionsNone Property**

CryptSharp

No options.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## **Syntax**

```
C# VB C++ F#

public static CrypterOptions None { get; }

Property Value
Type: CrypterOptions
```

#### **See Also**

Reference

**CrypterOptions Class** 

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# **CrypterProperty Class**

CryptSharp

Properties inherent to particular crypt algorithms.

#### **Inheritance Hierarchy**

SystemObject CryptSharpCrypterProperty

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

#### **Syntax**

```
C# VB C++ F#

Copy

public class CrypterProperty
```

The **CrypterProperty** type exposes the following members.

#### **Fields**

	Name	Description
₽ S	MaxPasswordLength	The maximum password length. Bytes beyond this length will have no effect.
<b>∮ S</b>	MaxRounds	The maximum number for Rounds.
<b>₽ S</b>	MinRounds	The minimum number for Rounds.

Top

#### See Also

Reference

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# **CrypterProperty Fields**

CryptSharp

The CrypterProperty type exposes the following members.

#### **Fields**

	Name	Description
₽ S	MaxPasswordLength	The maximum password length. Bytes beyond this length will have no effect.
S M	MaxRounds	The maximum number for Rounds.
₽ S	MinRounds	The minimum number for Rounds.

Top

## See Also

Reference

CrypterProperty Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger Send comments on this topic to jfb@zer7.com

## CrypterPropertyMaxPasswordLength Field CryptSharp

The maximum password length. Bytes beyond this length will have no effect.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**

C# VB C++ F#

Copy

public static readonly CrypterOptionKey MaxPasswordL

Field Value

Type: CrypterOptionKey

#### **See Also**

Reference

CrypterProperty Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# **CrypterPropertyMaxRounds Field**

CryptSharp

The maximum number for Rounds.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

#### **Syntax**

C# VB C++ F#

Copy

public static readonly CrypterOptionKey MaxRounds

Field Value

Type: CrypterOptionKey

## See Also

Reference

CrypterProperty Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# **CrypterPropertyMinRounds Field**

CryptSharp

The minimum number for Rounds.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

#### **Syntax**

C# VB C++ F#

Copy

public static readonly CrypterOptionKey MinRounds

Field Value

Type: CrypterOptionKey

#### See Also

Reference

CrypterProperty Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# **ExtendedDesCrypter Class**

CryptSharp

Extended DES crypt.

#### **Inheritance Hierarchy**

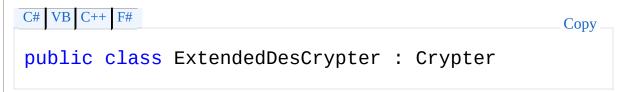
SystemObject CryptSharpCrypter

Crypt Sharp Extended Des Crypter

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

#### **Syntax**



The **ExtendedDesCrypter** type exposes the following members.

#### **Constructors**

	Name	Description
<b>=</b>	ExtendedDesCrypter	Initializes a new instance of the <b>ExtendedDesCrypter</b> class

Top

#### **Methods**

	Name	Description
<b>≡</b>	CanCrypt	Checks if the particular crypt algorithm is compatible with the salt string or crypted password.  (Overrides CrypterCanCrypt(String).)
<b>≡</b>	Crypt(Byte)	Creates a one-way password hash (crypted password) from password bytes.  (Inherited from Crypter.)
<b>≡</b>	Crypt(String)	Creates a one-way password hash (crypted password) from a password string. (Inherited from Crypter.)
<b>≡</b>	Crypt(Byte, CrypterOptions)	Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation. (Inherited from Crypter.)

<b>≟</b>	Crypt(String, CrypterOptions)	Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)
∉•	Crypt(String, String)	Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Inherited from Crypter.)
₫•	Crypt(Byte, String)	Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords.  (Overrides CrypterCrypt(Byte, String).)
<b>≓</b>	GenerateSalt	Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password (Inherited from Crypter.)
<b>∃</b>	GenerateSalt(Int32)	Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password.  (Inherited from Crypter.)
<b>∄</b>	GenerateSalt(CrypterOptions)	Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form algorithm+salt+hash, so the salt is always carried along with the crypted password). (Overrides CrypterGenerateSalt(CrypterOptions).)

Top

# **Properties**

Name	Description
_	Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys. (Overrides CrypterProperties.)

Top

# See Also

Reference

CryptSharp Namespace

# **ExtendedDesCrypter Constructor**

CryptSharp

Initializes a new instance of the ExtendedDesCrypter class

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**

```
C# VB C++ F#

public ExtendedDesCrypter()
```

## See Also

Reference

ExtendedDesCrypter Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# **ExtendedDesCrypter Methods**

The ExtendedDesCrypter type exposes the following members.

# Methods

	Name	Description
<b>≡</b>	CanCrypt	Checks if the particular crypt algorithm is compatible with the salt string or crypted password.  (Overrides CrypterCanCrypt(String).)
<b>≡</b>	Crypt(Byte)	Creates a one-way password hash (crypted password) from password bytes. (Inherited from Crypter.)
<b>≡</b>	Crypt(String)	Creates a one-way password hash (crypted password) from a password string. (Inherited from Crypter.)
<b>≡</b>	Crypt(Byte, CrypterOptions)	Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation. (Inherited from Crypter.)
<b>≡</b>	Crypt(String, CrypterOptions)	Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)
≡ 🍑	Crypt(String, String)	Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Inherited from Crypter.)
≅ <b>©</b>	Crypt(Byte, String)	Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Overrides CrypterCrypt(Byte, String).)

=	•	GenerateSalt	Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)
-	•	GenerateSalt(Int32)	Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password.  (Inherited from Crypter.)
=	•	GenerateSalt(CrypterOptions)	Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form algorithm+salt+hash, so the salt is always carried along with the crypted password). (Overrides CrypterGenerateSalt(CrypterOptions).)

Top

# **See Also**

Reference

ExtendedDesCrypter Class CryptSharp Namespace

# **ExtendedDesCrypterCanCrypt Method**

CryptSharp

Checks if the particular crypt algorithm is compatible with the salt string or crypted password.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

```
public override bool CanCrypt(
    string salt
)

Parameters
salt
    Type: SystemString
    The salt string or crypted password.

Return Value
Type: Boolean
true if the algorithm is compatible.

See Also

Reference
ExtendedDesCrypter Class
```

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com

CryptSharp Namespace

# **Overload List**

	Name	Description	
<b>≡</b>	Crypt(Byte)	Creates a one-way password hash (crypted password) from password bytes. (Inherited from Crypter.)	
<b>≡</b>	Crypt(String)	Creates a one-way password hash (crypted password) from a password string.  (Inherited from Crypter.)	
<b>≡</b>	Crypt(Byte, CrypterOptions)	Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation. (Inherited from Crypter.)	
<b>≡</b>	Crypt(String, CrypterOptions)	Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)	
=0	Crypt(String, String)	Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Inherited from Crypter.)	
=0	Crypt(Byte, String)	Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Overrides CrypterCrypt(Byte, String).)	

Top

# **See Also**

#### Reference

ExtendedDesCrypter Class CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# ExtendedDesCrypterCrypt Method (Byte, String)

CryptSharp

Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using

GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate \*and\* verify crypted passwords.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## **Syntax**

salt

Type: SystemString

The salt string or crypted password containing a salt string.

**Return Value** 

Type: String

The crypted password.

#### **See Also**

#### Reference

ExtendedDesCrypter Class

**Crypt Overload** 

# CryptSharp Namespace

# $Extended Des Crypter Generate Salt\ Method \\ \hbox{\it CryptSharp}$

# **Overload List**

	Name	Description
=	GenerateSalt	Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)
=₩	GenerateSalt(Int32)	Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password.  (Inherited from Crypter.)
₫ 🕸	GenerateSalt(CrypterOptions)	Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form algorithm+salt+hash, so the salt is always carried along with the crypted password). (Overrides CrypterGenerateSalt(CrypterOptions).)

Top

# **See Also**

Reference

ExtendedDesCrypter Class CryptSharp Namespace

# ExtendedDesCrypterGenerateSalt Method (CrypterOptions)

CryptSharp

Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form algorithm+salt+hash, so the salt is always carried along with the crypted password).

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## **Syntax**

#### **Parameters**

options

Type: CryptSharpCrypterOptions

Options modifying the salt generation.

#### **Return Value**

Type: String
The salt string.

#### See Also

#### Reference

ExtendedDesCrypter Class

GenerateSalt Overload

CryptSharp Namespace

# **ExtendedDesCrypter Properties**

CryptSharp

The ExtendedDesCrypter type exposes the following members.

# **Properties**

Name	Description
Properties	Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys. (Overrides CrypterProperties.)

Top

# **See Also**

Reference

ExtendedDesCrypter Class CryptSharp Namespace

# ExtendedDesCrypterProperties Property CryptSharp

Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**



# **LdapCrypter Class**

CryptSharp

LDAP schemes such as {SHA}.

## **Inheritance Hierarchy**

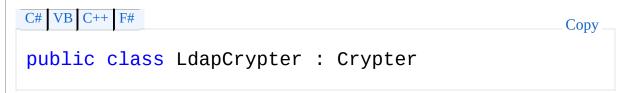
SystemObject CryptSharpCrypter

Crypt Sharp Ldap Crypter

Namespace: CryptSharp

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**



The **LdapCrypter** type exposes the following members.

#### **Constructors**

	Name	Description
= (	LdapCrypter	Initializes a new instance of the <b>LdapCrypter</b> class. The specified environment is searched for the {CRYPT} password scheme.

Top

#### **Methods**

	Name	Description
<b>≡</b>	CanCrypt	Checks if the particular crypt algorithm is compatible with the salt string or crypted password.  (Overrides CrypterCanCrypt(String).)
<b>≡</b>	Crypt(Byte)	Creates a one-way password hash (crypted password) from password bytes. (Inherited from Crypter.)
<b>≡</b>	Crypt(String)	Creates a one-way password hash (crypted password) from a password string. (Inherited from Crypter.)
<b>≡</b>	Crypt(Byte, CrypterOptions)	Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation.

		(Inherited from Crypter.)
<b>≡</b>	Crypt(String, CrypterOptions)	Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)
=	Crypt(String, String)	Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Inherited from Crypter.)
=	Crypt(Byte, String)	Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords.  (Overrides CrypterCrypt(Byte, String).)
<b>≡</b>	GenerateSalt	Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)
<b>≡</b>	GenerateSalt(Int32)	Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)
<b>≡</b>	GenerateSalt(CrypterOptions)	Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form algorithm+salt+hash, so the salt is always carried

along with the crypted password). (Overrides CrypterGenerateSalt(CrypterOptions).)

#### Top

# **Properties**

Name	Description
Properties	Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys. (Inherited from Crypter.)

#### Top

# **See Also**

#### Reference

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# **LdapCrypter Constructor**

CryptSharp

Initializes a new instance of the LdapCrypter class. The specified environment is searched for the {CRYPT} password scheme.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

#### **Parameters**

*cryptSchemeEnvironment* 

Type: CryptSharpCrypterEnvironment

The environment for the {CRYPT} passwod scheme.

### **See Also**

Reference

LdapCrypter Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# **LdapCrypter Methods**

The LdapCrypter type exposes the following members.

# Methods

	Name	Description
<b>≡</b>	CanCrypt	Checks if the particular crypt algorithm is compatible with the salt string or crypted password. (Overrides CrypterCanCrypt(String).)
<b>≡</b>	Crypt(Byte)	Creates a one-way password hash (crypted password) from password bytes. (Inherited from Crypter.)
<b>≡</b>	Crypt(String)	Creates a one-way password hash (crypted password) from a password string. (Inherited from Crypter.)
<b>≡</b>	Crypt(Byte, CrypterOptions)	Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation. (Inherited from Crypter.)
<b>≡</b>	Crypt(String, CrypterOptions)	Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)
∃	Crypt(String, String)	Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Inherited from Crypter.)
€ 🚱	Crypt(Byte, String)	Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Overrides CrypterCrypt(Byte, String).)

=	•	GenerateSalt	Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)
	•	GenerateSalt(Int32)	Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password.  (Inherited from Crypter.)
=	•	GenerateSalt(CrypterOptions)	Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form algorithm+salt+hash, so the salt is always carried along with the crypted password). (Overrides CrypterGenerateSalt(CrypterOptions).)

Top

# **See Also**

Reference

LdapCrypter Class CryptSharp Namespace

# LdapCrypterCanCrypt Method

CryptSharp

Checks if the particular crypt algorithm is compatible with the salt string or crypted password.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com

CryptSharp Namespace

# **Overload List**

	Name	Description	
<b>≡</b>	Crypt(Byte)	Creates a one-way password hash (crypted password) from password bytes. (Inherited from Crypter.)	
<b>≡</b>	Crypt(String)	Creates a one-way password hash (crypted password) from a password string. (Inherited from Crypter.)	
<b>≡</b>	Crypt(Byte, CrypterOptions)	Creates a one-way password hash (crypted password) from password bytes.  Options modify the crypt operation.  (Inherited from Crypter.)	
<b>≡</b>	Crypt(String, CrypterOptions)	Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)	
≡₩	Crypt(String, String)	Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Inherited from Crypter.)	
and a salt string. The salt can be produced using  GenerateSalt(CrypterOptions). Because crypted password  algorithm+salt+hash, if you pass a crypted password  parameter, the same algorithm and salt will be used to re-  password. Since randomness comes from the salt, the same  same hash, and so the same crypted password will result.		GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords.	

Top

# **See Also**

#### Reference

LdapCrypter Class CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

## LdapCrypterCrypt Method (Byte, String) CryptSharp

Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using <code>GenerateSalt(CrypterOptions)</code>. Because crypted passwords take the form <code>algorithm+salt+hash</code>, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate \*and\* verify crypted passwords.

Namespace: CryptSharp

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## **Syntax**

```
C# VB C++ F#
                                                                 Copy
 public override string Crypt(
            byte[] password,
            string salt
 )
Parameters
password
  Type: SystemByte
  The bytes of the password.
salt
  Type: SystemString
  The salt string or crypted password containing a salt string.
Return Value
Type: String
The crypted password.
```

#### **See Also**

Reference

LdapCrypter Class
Crypt Overload

# CryptSharp Namespace

# **Overload List**

	Name	Description
=	GenerateSalt	Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)
Ξ₩	GenerateSalt(Int32)	Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password.  (Inherited from Crypter.)
	GenerateSalt(CrypterOptions)	Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form algorithm+salt+hash, so the salt is always carried along with the crypted password). (Overrides CrypterGenerateSalt(CrypterOptions).)

Top

# See Also

Reference

LdapCrypter Class CryptSharp Namespace

# LdapCrypterGenerateSalt Method (CrypterOptions)

CryptSharp

Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form algorithm+salt+hash, so the salt is always carried along with the crypted password).

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## **Syntax**

#### **Parameters**

options

Type: CryptSharpCrypterOptions

Options modifying the salt generation.

#### **Return Value**

Type: String
The salt string.

#### See Also

#### Reference

LdapCrypter Class

GenerateSalt Overload

CryptSharp Namespace

# **LdapCrypter Properties**

CryptSharp

The LdapCrypter type exposes the following members.

# **Properties**

Name	Description
Properties	Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys. (Inherited from Crypter.)

Top

# See Also

Reference

LdapCrypter Class CryptSharp Namespace

# **LdapCrypterOption Class**

CryptSharp

Options that modify the LDAP crypt operation.

# **Inheritance Hierarchy**

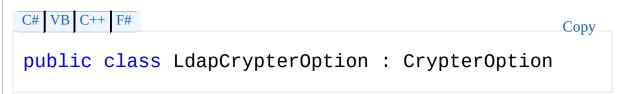
SystemObject CryptSharpCrypterOption

Crypt Sharp Ldap Crypter Option

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**



The **LdapCrypterOption** type exposes the following members.

#### **Fields**

		Name	Description
	<b>₽ S</b>	Crypter	The crypter to use with Crypt.
<b>⋄ S</b> CrypterOptions		CrypterOptions	The options to pass to the crypter specified by Crypter.

Top

#### **See Also**

Reference

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# **LdapCrypterOption Fields**

CryptSharp

The LdapCrypterOption type exposes the following members.

## **Fields**

	Name	Description
øs	Crypter	The crypter to use with Crypt.
<b>₽ S</b>	CrypterOptions	The options to pass to the crypter specified by Crypter.

#### Top

# See Also

#### Reference

LdapCrypterOption Class CryptSharp Namespace

# LdapCrypterOptionCrypter Field

CryptSharp

The crypter to use with Crypt.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**

C# VB C++ F#

Copy

public static readonly CrypterOptionKey Crypter

Field Value

Type: CrypterOptionKey

#### **See Also**

Reference

LdapCrypterOption Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# LdapCrypterOptionCrypterOptions Field CryptSharp

The options to pass to the crypter specified by Crypter.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## **Syntax**

C# VB C++ F#

Copy

public static readonly CrypterOptionKey CrypterOptio

Field Value

Type: CrypterOptionKey

#### **See Also**

Reference

 $Ldap Crypter Option\ Class$ 

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# **LdapCrypterVariant Enumeration**

CryptSharp

LDAP password schemes.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**

C# VB C++ F#

Copy

public enum LdapCrypterVariant

#### **Members**

Member name	Value	Description
SSha	0	Salted SHA-1. This is the default.
Sha	1	Unsalted SHA-1. Used in htpasswd files.
SSha256	6	Salted SHA-256.
Sha256	7	Unsalted SHA-256.
SSha384	8	Salted SHA-384.
Sha384	9	Unsalted SHA-384.
SSha512	10	Salted SHA-512.
Sha512	11	Unsalted SHA-512.
SMD5	2	Salted MD-5.
MD5	3	Unsalted MD-5.
Cleartext	4	No crypt operation is performed. The password can be read easily.
Crypt	5	Any crypt algorithm. If you specify this for Variant, you must also set Crypter and may optionally set CrypterOptions.

#### **See Also**

Reference

CryptSharp Namespace

# **MD5Crypter Class**

CryptSharp

MD5 crypt, supported by nearly all systems. A variant supports Apache htpasswd files.

# **Inheritance Hierarchy**

SystemObject CryptSharpCrypter

CryptSharpMD5Crypter

Namespace: CryptSharp

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## **Syntax**

```
C# VB C++ F#

Copy

public class MD5Crypter : Crypter
```

The **MD5Crypter** type exposes the following members.

#### **Constructors**

	Name	Description
<b>≡</b>	MD5Crypter	Initializes a new instance of the MD5Crypter class

Top

#### **Methods**

	Name	Description
	rame	Description
<b>≡</b>	CanCrypt	Checks if the particular crypt algorithm is compatible with the salt string or crypted password. (Overrides CrypterCanCrypt(String).)
<b>=</b>	Crypt(Byte)	Creates a one-way password hash (crypted password) from password bytes.  (Inherited from Crypter.)
<b>≡</b>	Crypt(String)	Creates a one-way password hash (crypted password) from a password string. (Inherited from Crypter.)
<b>≡</b>	Crypt(Byte, CrypterOptions)	Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation.

		(Inherited from Crypter.)
<b>≡</b>	Crypt(String, CrypterOptions)	Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)
=	Crypt(String, String)	Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Inherited from Crypter.)
=	Crypt(Byte, String)	Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords.  (Overrides CrypterCrypt(Byte, String).)
<b>≡</b>	GenerateSalt	Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)
<b>≡</b>	GenerateSalt(Int32)	Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)
<b>≡</b>	GenerateSalt(CrypterOptions)	Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form algorithm+salt+hash, so the salt is always carried

llong with the crypted password).	
Overrides CrypterGenerateSalt(CrypterOption	s).)

#### Top

# **Properties**

Name	Description
Properties	Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys. (Inherited from Crypter.)

### Top

# **See Also**

#### Reference

CryptSharp Namespace

# **MD5Crypter Constructor**

CryptSharp

Initializes a new instance of the MD5Crypter class

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**

```
C# VB C++ F#

Copy

public MD5Crypter()
```

### See Also

Reference

**MD5Crypter Class** 

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# **MD5Crypter Methods**

The MD5Crypter type exposes the following members.

# Methods

	Name	Description
<b>≡</b>	CanCrypt	Checks if the particular crypt algorithm is compatible with the salt string or crypted password. (Overrides CrypterCanCrypt(String).)
<b>≡</b>	Crypt(Byte)	Creates a one-way password hash (crypted password) from password bytes. (Inherited from Crypter.)
<b>≡</b>	Crypt(String)	Creates a one-way password hash (crypted password) from a password string. (Inherited from Crypter.)
<b>≡</b>	Crypt(Byte, CrypterOptions)	Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation. (Inherited from Crypter.)
<b>≡</b>	Crypt(String, CrypterOptions)	Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)
∃	Crypt(String, String)	Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Inherited from Crypter.)
€ 🚱	Crypt(Byte, String)	Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Overrides CrypterCrypt(Byte, String).)

<b>≡</b>	GenerateSalt	Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)
Ξ₩	GenerateSalt(Int32)	Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password.  (Inherited from Crypter.)
≡₩	GenerateSalt(CrypterOptions)	Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form algorithm+salt+hash, so the salt is always carried along with the crypted password). (Overrides CrypterGenerateSalt(CrypterOptions).)

Top

# **See Also**

Reference

MD5Crypter Class CryptSharp Namespace

# MD5CrypterCanCrypt Method

CryptSharp

Checks if the particular crypt algorithm is compatible with the salt string or crypted password.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

#### **Syntax**

```
public override bool CanCrypt(
    string salt
)

Parameters
salt
    Type: SystemString
    The salt string or crypted password.

Return Value
Type: Boolean
true if the algorithm is compatible.

See Also
Reference
MD5Crypter Class
```

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com

CryptSharp Namespace

# **Overload List**

	Name	Description
<b>≡</b>	Crypt(Byte)	Creates a one-way password hash (crypted password) from password bytes. (Inherited from Crypter.)
<b>≡</b>	Crypt(String)	Creates a one-way password hash (crypted password) from a password string.  (Inherited from Crypter.)
<b>≡</b>	Crypt(Byte, CrypterOptions)	Creates a one-way password hash (crypted password) from password bytes.  Options modify the crypt operation.  (Inherited from Crypter.)
<b>≡</b>	Crypt(String, CrypterOptions)	Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)
=0	Crypt(String, String)	Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Inherited from Crypter.)
=0	Crypt(Byte, String)	Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Overrides CrypterCrypt(Byte, String).)

Top

# **See Also**

Reference

MD5Crypter Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

#### MD5CrypterCrypt Method (Byte, String) CryptSharp

Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate \*and\* verify crypted passwords.

Namespace: CryptSharp

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

```
C# VB C++ F#
                                                                 Copy
 public override string Crypt(
            byte[] password,
            string salt
 )
Parameters
password
  Type: SystemByte
  The bytes of the password.
salt
  Type: SystemString
  The salt string or crypted password containing a salt string.
Return Value
Type: String
The crypted password.
```

#### See Also

Reference

MD5Crypter Class

**Crypt Overload** 

# CryptSharp Namespace

# **Overload List**

	Name	Description
=	GenerateSalt	Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)
Ξ₩	GenerateSalt(Int32)	Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password.  (Inherited from Crypter.)
	GenerateSalt(CrypterOptions)	Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form algorithm+salt+hash, so the salt is always carried along with the crypted password). (Overrides CrypterGenerateSalt(CrypterOptions).)

Top

# **See Also**

Reference

MD5Crypter Class

CryptSharp Namespace

# MD5CrypterGenerateSalt Method (CrypterOptions)

CryptSharp

Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form algorithm+salt+hash, so the salt is always carried along with the crypted password).

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

#### **Syntax**

```
C# VB C++ F#
                                                      Copy
public override string GenerateSalt(
          CrypterOptions options
Parameters
```

options

Type: CryptSharpCrypterOptions

Options modifying the salt generation.

#### **Return Value**

Type: String The salt string.

#### See Also

#### Reference

**MD5Crypter Class** 

GenerateSalt Overload

CryptSharp Namespace

# **MD5Crypter Properties**

CryptSharp

The MD5Crypter type exposes the following members.

# **Properties**

Name	Description
Properties	Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys. (Inherited from Crypter.)

Top

# See Also

Reference

MD5Crypter Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# **MD5CrypterVariant Enumeration**

CryptSharp

Modified versions of the MD5 crypt algorithm.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

C# VB C++ F#

Copy

public enum MD5CrypterVariant

#### **Members**

Member name	Value	Description
Standard	0	Standard MD5 crypt.
Apache	1	Apache htpasswd files have a different prefix. Due to the nature of MD5 crypt, this also affects the crypted password.

#### **See Also**

Reference

CryptSharp Namespace

# **PhpassCrypter Class**

CryptSharp

PHPass crypt. Used by WordPress. Variants support phpBB and Drupal 7+.

### **Inheritance Hierarchy**

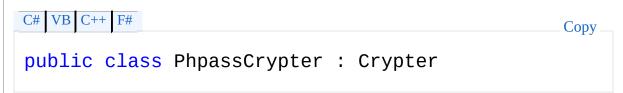
SystemObject CryptSharpCrypter

CryptSharpPhpassCrypter

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

#### **Syntax**



The **PhpassCrypter** type exposes the following members.

#### **Constructors**

	Name	Description
<b>≡</b>	PhpassCrypter	Initializes a new instance of the <b>PhpassCrypter</b> class

Top

#### **Methods**

	Name	Description
=0	CanCrypt	Checks if the particular crypt algorithm is compatible with the salt string or crypted password.  (Overrides CrypterCanCrypt(String).)
<b>≡</b>	Crypt(Byte)	Creates a one-way password hash (crypted password) from password bytes. (Inherited from Crypter.)
<b>≡</b>	Crypt(String)	Creates a one-way password hash (crypted password) from a password string. (Inherited from Crypter.)
<b>≡</b>	Crypt(Byte, CrypterOptions)	Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation. (Inherited from Crypter.)

<b>≟</b>	Crypt(String, CrypterOptions)	Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)
∉•	Crypt(String, String)	Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Inherited from Crypter.)
₫•	Crypt(Byte, String)	Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords.  (Overrides CrypterCrypt(Byte, String).)
<b>≓</b>	GenerateSalt	Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password (Inherited from Crypter.)
<b>∃</b>	GenerateSalt(Int32)	Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password.  (Inherited from Crypter.)
<b>∄</b>	GenerateSalt(CrypterOptions)	Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form algorithm+salt+hash, so the salt is always carried along with the crypted password). (Overrides CrypterGenerateSalt(CrypterOptions).)

Top

# **Properties**

Name	Description
Properties	Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys. (Overrides CrypterProperties.)

Top

# See Also

Reference

CryptSharp Namespace

# **PhpassCrypter Constructor**

CryptSharp

Initializes a new instance of the PhpassCrypter class

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**

```
C# VB C++ F#

public PhpassCrypter()
```

### See Also

Reference

PhpassCrypter Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# **PhpassCrypter Methods**

The PhpassCrypter type exposes the following members.

# Methods

	Name	Description
<b>≡</b>	CanCrypt	Checks if the particular crypt algorithm is compatible with the salt string or crypted password.  (Overrides CrypterCanCrypt(String).)
<b>≡</b>	Crypt(Byte)	Creates a one-way password hash (crypted password) from password bytes. (Inherited from Crypter.)
<b>≡</b>	Crypt(String)	Creates a one-way password hash (crypted password) from a password string. (Inherited from Crypter.)
<b>≡</b>	Crypt(Byte, CrypterOptions)	Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation. (Inherited from Crypter.)
<b>≡</b>	Crypt(String, CrypterOptions)	Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)
≡ 🍑	Crypt(String, String)	Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Inherited from Crypter.)
≅ <b>©</b>	Crypt(Byte, String)	Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Overrides CrypterCrypt(Byte, String).)

=	•	GenerateSalt	Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)
-	•	GenerateSalt(Int32)	Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password.  (Inherited from Crypter.)
=	•	GenerateSalt(CrypterOptions)	Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form algorithm+salt+hash, so the salt is always carried along with the crypted password). (Overrides CrypterGenerateSalt(CrypterOptions).)

Top

# **See Also**

Reference

PhpassCrypter Class CryptSharp Namespace

# PhpassCrypterCanCrypt Method

CryptSharp

Checks if the particular crypt algorithm is compatible with the salt string or crypted password.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

#### **Syntax**

```
public override bool CanCrypt(
    string salt
)

Parameters
salt
    Type: SystemString
    The salt string or crypted password.

Return Value
Type: Boolean
true if the algorithm is compatible.

See Also

Reference
PhpassCrypter Class
```

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com

CryptSharp Namespace

# **Overload List**

	Name	Description
<b>≡</b>	Crypt(Byte)	Creates a one-way password hash (crypted password) from password bytes. (Inherited from Crypter.)
string.		Creates a one-way password hash (crypted password) from a password string. (Inherited from Crypter.)
<b>≡</b>	Crypt(Byte, CrypterOptions)	Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation. (Inherited from Crypter.)
<b>≡</b>	Crypt(String, CrypterOptions)	Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)
≡ <b>◊</b>	Crypt(String, String)	Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Inherited from Crypter.)
ΞQ	Crypt(Byte, String)	Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using <a href="GenerateSalt(CrypterOptions">GenerateSalt(CrypterOptions</a> ). Because crypted passwords take the form <a href="algorithm+salt+hash">algorithm+salt+hash</a> , if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Overrides CrypterCrypt(Byte, String).)

Top

# **See Also**

#### Reference

PhpassCrypter Class CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

### PhpassCrypterCrypt Method (Byte, String) CryptSharp

Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using

GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate \*and\* verify crypted passwords.

Namespace: CryptSharp

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**

```
C# VB C++ F#
                                                            Copy
 public override string Crypt(
           byte[] password,
           string salt
 )
Parameters
password
  Type: SystemByte
  The bytes of the password.
salt
```

Type: SystemString

The salt string or crypted password containing a salt string.

#### **Return Value**

Type: String

The crypted password.

#### See Also

#### Reference

PhpassCrypter Class

**Crypt Overload** 

# CryptSharp Namespace

# **Overload List**

	Name	Description
=	GenerateSalt	Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)
€₩	GenerateSalt(Int32)	Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password.  (Inherited from Crypter.)
≡ 🍑	GenerateSalt(CrypterOptions)	Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form algorithm+salt+hash, so the salt is always carried along with the crypted password). (Overrides CrypterGenerateSalt(CrypterOptions).)

Top

# **See Also**

Reference

PhpassCrypter Class CryptSharp Namespace

# PhpassCrypterGenerateSalt Method (CrypterOptions)

CryptSharp

Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form algorithm+salt+hash, so the salt is always carried along with the crypted password).

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

```
C# VB C++ F#
                                                      Copy
public override string GenerateSalt(
          CrypterOptions options
Parameters
```

options

Type: CryptSharpCrypterOptions

Options modifying the salt generation.

#### **Return Value**

Type: String The salt string.

#### See Also

#### Reference

PhpassCrypter Class

GenerateSalt Overload

CryptSharp Namespace

# **PhpassCrypter Properties**

CryptSharp

The PhpassCrypter type exposes the following members.

# **Properties**

Name	Description
•	Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys. (Overrides CrypterProperties.)

Top

# **See Also**

Reference

PhpassCrypter Class CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# **PhpassCrypterProperties Property**

CryptSharp

Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys.

Namespace: CryptSharp

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

#### **Syntax**



# **PhpassCrypterVariant Enumeration**

CryptSharp

Modified versions of the PHPass crypt algorithm.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

C# VB C++ F# Copy\_
public enum PhpassCrypterVariant

#### **Members**

Member name	Value	Description
Standard	0	Standard PHPass. WordPress uses this.
Phpbb	1	phpBB changes the prefix but the algorithm is otherwise identical.
Drupal	2	Drupal 7+ uses SHA512 instead of MD5.

#### See Also

Reference

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# **Sha256Crypter Class**

CryptSharp

SHA256 crypt. A reasonable choice if you cannot use Blowfish crypt for policy reasons.

### **Inheritance Hierarchy**

SystemObject CryptSharpCrypter CryptSharpShaCrypter

CryptSharpSha256Crypter

Namespace: CryptSharp

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

C# VB C++ F#

Copy

public class Sha256Crypter : ShaCrypter

The **Sha256Crypter** type exposes the following members.

#### **Constructors**

	Name	Description
<b>=</b>	Sha256Crypter	Initializes a new instance of the <b>Sha256Crypter</b> class

Top

#### **Methods**

	Name	Description
<b>≡</b>	CanCrypt	Checks if the particular crypt algorithm is compatible with the salt string or crypted password.  (Inherited from ShaCrypter.)
<b>≟</b>	Crypt(Byte)	Creates a one-way password hash (crypted password) from password bytes. (Inherited from Crypter.)
=0	Crypt(String)	Creates a one-way password hash (crypted password) from a password string. (Inherited from Crypter.)

	<b>≟</b>	Crypt(Byte, CrypterOptions)	Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation. (Inherited from Crypter.)
	<b>≟</b>	Crypt(String, CrypterOptions)	Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)
	€ 🗳	Crypt(String, String)	Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Inherited from Crypter.)
	=₩	Crypt(Byte, String)	Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Inherited from ShaCrypter.)
	<b>≡</b>	GenerateSalt	Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)
	<b>≟</b>	GenerateSalt(Int32)	Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password.  (Inherited from Crypter.)
7.	≅ 🍑	GenerateSalt(CrypterOptions)	Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different

crypted password (crypted passwords have the form		
algorithm+salt+hash, so the salt is always carried		
along with the crypted password).		
(Inherited from ShaCrypter.)		

#### Top

# **Properties**

Name	Description
Properties	Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys. (Inherited from ShaCrypter.)

#### Top

# See Also

Reference

CryptSharp Namespace

# **Sha256Crypter Constructor**

CryptSharp

Initializes a new instance of the Sha256Crypter class

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**

```
C# VB C++ F#

Copy

public Sha256Crypter()
```

### See Also

Reference

Sha256Crypter Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

# **Sha256Crypter Methods**

The Sha256Crypter type exposes the following members.

	Name	Description
<b>≡</b>	CanCrypt	Checks if the particular crypt algorithm is compatible with the salt string or crypted password.  (Inherited from ShaCrypter.)
<b>≡</b>	Crypt(Byte)	Creates a one-way password hash (crypted password) from password bytes. (Inherited from Crypter.)
<b>≡</b>	Crypt(String)	Creates a one-way password hash (crypted password) from a password string. (Inherited from Crypter.)
<b>≡</b>	Crypt(Byte, CrypterOptions)	Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation. (Inherited from Crypter.)
<b>≡</b>	Crypt(String, CrypterOptions)	Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)
≡ <b>`</b>	Crypt(String, String)	Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Inherited from Crypter.)
€₩	Crypt(Byte, String)	Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Inherited from ShaCrypter.)

=	GenerateSalt	Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)
=0	GenerateSalt(Int32)	Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password.  (Inherited from Crypter.)
⊒••	GenerateSalt(CrypterOptions)	Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form algorithm+salt+hash, so the salt is always carried along with the crypted password). (Inherited from ShaCrypter.)

### **See Also**

Reference

Sha256Crypter Class CryptSharp Namespace

# **Sha256Crypter Properties**

CryptSharp

The Sha256Crypter type exposes the following members.

# **Properties**

Name	Description
Properties	Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys. (Inherited from ShaCrypter.)

Top

### See Also

Reference

Sha256Crypter Class CryptSharp Namespace

### **Sha512Crypter Class**

CryptSharp

SHA512 crypt. A reasonable choice if you cannot use Blowfish crypt for policy reasons.

## **Inheritance Hierarchy**

SystemObject CryptSharpCrypter CryptSharpShaCrypter

CryptSharpSha512Crypter

Namespace: CryptSharp

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

C# VB C++ F#

Copy

public class Sha512Crypter : ShaCrypter

The **Sha512Crypter** type exposes the following members.

### **Constructors**

	Name	Description
<b>≡</b>	Sha512Crypter	Initializes a new instance of the <b>Sha512Crypter</b> class

Top

ame	Description
Sã	Checks if the particular crypt algorithm is compatible with the salt string or crypted password.  (Inherited from ShaCrypter.)
pe	Creates a one-way password hash (crypted password) from password bytes.  (Inherited from Crypter.)
pa	Creates a one-way password hash (crypted password) from a password string. (Inherited from Crypter.)
pa	password string.

	<b>≟</b>	Crypt(Byte, CrypterOptions)	Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation. (Inherited from Crypter.)
	<b>≟</b>	Crypt(String, CrypterOptions)	Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)
	€ 🍑	Crypt(String, String)	Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Inherited from Crypter.)
	=₩	Crypt(Byte, String)	Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Inherited from ShaCrypter.)
	<b>≡</b>	GenerateSalt	Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)
	<b>≟</b>	GenerateSalt(Int32)	Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password.  (Inherited from Crypter.)
	≅ 🍑	GenerateSalt(CrypterOptions)	Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different

crypted password (crypted passwords have the form	
algorithm+salt+hash, so the salt is always carried	
along with the crypted password).	
(Inherited from ShaCrypter.)	

# **Properties**

Name	Description
Properties	Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys. (Inherited from ShaCrypter.)

### Top

# See Also

Reference

CryptSharp Namespace

# **Sha512Crypter Constructor**

CryptSharp

Initializes a new instance of the Sha512Crypter class

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

```
C# VB C++ F#

Copy

public Sha512Crypter()
```

### **See Also**

Reference

Sha512Crypter Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com

# **Sha512Crypter Methods**

The Sha512Crypter type exposes the following members.

	Name	Description
<b>≡</b>	CanCrypt	Checks if the particular crypt algorithm is compatible with the salt string or crypted password.  (Inherited from ShaCrypter.)
<b>≡</b>	Crypt(Byte)	Creates a one-way password hash (crypted password) from password bytes. (Inherited from Crypter.)
<b>≡</b>	Crypt(String)	Creates a one-way password hash (crypted password) from a password string. (Inherited from Crypter.)
<b>≡</b>	Crypt(Byte, CrypterOptions)	Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation. (Inherited from Crypter.)
<b>≡</b>	Crypt(String, CrypterOptions)	Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)
≡ <b>`</b>	Crypt(String, String)	Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Inherited from Crypter.)
€₩	Crypt(Byte, String)	Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Inherited from ShaCrypter.)

	=•	GenerateSalt	Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)
	=	GenerateSalt(Int32)	Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password.  (Inherited from Crypter.)
	₫ 🍑	GenerateSalt(CrypterOptions)	Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form algorithm+salt+hash, so the salt is always carried along with the crypted password). (Inherited from ShaCrypter.)

### **See Also**

Reference

Sha512Crypter Class CryptSharp Namespace

# **Sha512Crypter Properties**

CryptSharp

The Sha512Crypter type exposes the following members.

# **Properties**

Name	Description
Properties	Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys. (Inherited from ShaCrypter.)

Top

### See Also

Reference

Sha512Crypter Class CryptSharp Namespace

CryptSharp

Base class for Sha256Crypter and Sha512Crypter.

### **Inheritance Hierarchy**

SystemObject CryptSharpCrypter

CryptSharpShaCrypter

CryptSharpSha256Crypter CryptSharpSha512Crypter

Namespace: CryptSharp

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

C# VB C++ F#

Copy

public abstract class ShaCrypter : Crypter

The **ShaCrypter** type exposes the following members.

	Name	Description
<b>∉</b>	CanCrypt	Checks if the particular crypt algorithm is compatible with the salt string or crypted password.  (Overrides CrypterCanCrypt(String).)
<b>∉</b>	Crypt(Byte)	Creates a one-way password hash (crypted password) from password bytes. (Inherited from Crypter.)
<b>∉</b>	Crypt(String)	Creates a one-way password hash (crypted password) from a password string. (Inherited from Crypter.)
<b>≡</b>	Crypt(Byte, CrypterOptions)	Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation. (Inherited from Crypter.)
<b>=</b>	Crypt(String, CrypterOptions)	Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)
<b>≡</b>	Crypt(String, String)	Creates a one-way password hash (crypted password) from a

			password string and a salt string. The salt can be produced using <code>GenerateSalt(CrypterOptions)</code> . Because crypted passwords take the form <code>algorithm+salt+hash</code> , if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Inherited from <code>Crypter</code> .)
	≘₩	Crypt(Byte, String)	Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Overrides CrypterCrypt(Byte, String).)
	<b>≡</b>	GenerateSalt	Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)
	<b>≡</b>	GenerateSalt(Int32)	Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password.  (Inherited from Crypter.)
	≅❖	GenerateSalt(CrypterOptions)	Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form algorithm+salt+hash, so the salt is always carried along with the crypted password). (Overrides CrypterGenerateSalt(CrypterOptions).)

# **Properties**

Name	Description
Properties	Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys. (Overrides CrypterProperties.)

# See Also

Reference

CryptSharp Namespace

# **ShaCrypter Methods**

The ShaCrypter type exposes the following members.

	Name	Description
<b>≡</b>	CanCrypt	Checks if the particular crypt algorithm is compatible with the salt string or crypted password. (Overrides CrypterCanCrypt(String).)
<b>≡</b>	Crypt(Byte)	Creates a one-way password hash (crypted password) from password bytes.  (Inherited from Crypter.)
<b>≡</b>	Crypt(String)	Creates a one-way password hash (crypted password) from a password string. (Inherited from Crypter.)
<b>≡</b>	Crypt(Byte, CrypterOptions)	Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation. (Inherited from Crypter.)
<b>≡</b>	Crypt(String, CrypterOptions)	Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)
∃	Crypt(String, String)	Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Inherited from Crypter.)
€ 🚱	Crypt(Byte, String)	Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Overrides CrypterCrypt(Byte, String).)

<b>≡</b>	GenerateSalt	Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)
Ξ₩	GenerateSalt(Int32)	Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password.  (Inherited from Crypter.)
≡₩	GenerateSalt(CrypterOptions)	Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form algorithm+salt+hash, so the salt is always carried along with the crypted password). (Overrides CrypterGenerateSalt(CrypterOptions).)

## **See Also**

Reference

ShaCrypter Class CryptSharp Namespace

### **ShaCrypterCanCrypt Method**

CryptSharp

Checks if the particular crypt algorithm is compatible with the salt string or crypted password.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

```
C# VB C++ F#
                                                                Copy
 public override bool CanCrypt(
            string salt
 )
Parameters
salt
  Type: SystemString
  The salt string or crypted password.
Return Value
Type: Boolean
true if the algorithm is compatible.
See Also
Reference
ShaCrypter Class
CryptSharp Namespace
```

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com

# **Overload List**

	Name	Description	
<b>≡</b>	Crypt(Byte)	Creates a one-way password hash (crypted password) from password bytes. (Inherited from Crypter.)	
<b>≡</b>	Crypt(String)	Creates a one-way password hash (crypted password) from a password string. (Inherited from Crypter.)	
Crypt(Byte, CrypterOptions)  Creates a one-way password hash (crypted password) from password password)  CrypterOptions Options modify the crypt operation.  (Inherited from Crypter.)			
<b>≡</b>	Crypt(String, CrypterOptions)  Creates a one-way password hash (crypted password) from a password tring. Options modify the crypt operation. (Inherited from Crypter.)		
<b>≅</b>	Crypt(String, String)	Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Inherited from Crypter.)	
Crypt(Byte, String)  Creates a one-way password hash (crypted password and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted pass algorithm+salt+hash, if you pass a crypted password. Since randomness comes from the salt, the same hash, and so the same crypted password will re		GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords.	

Top

## **See Also**

Reference

ShaCrypter Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com

### **ShaCrypterCrypt Method (Byte, String)**

CryptSharp

Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using

GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate \*and\* verify crypted passwords.

Namespace: CryptSharp

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

```
C# VB C++ F#
                                                                 Copy
 public override string Crypt(
            byte[] password,
            string salt
 )
Parameters
password
  Type: SystemByte
  The bytes of the password.
salt
  Type: SystemString
  The salt string or crypted password containing a salt string.
Return Value
```

Type: String

The crypted password.

### See Also

#### Reference

ShaCrypter Class

**Crypt Overload** 

# CryptSharp Namespace

### **Overload List**

	Name	Description
=	GenerateSalt	Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)
Ξ₩	GenerateSalt(Int32)	Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password.  (Inherited from Crypter.)
	GenerateSalt(CrypterOptions)	Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form algorithm+salt+hash, so the salt is always carried along with the crypted password). (Overrides CrypterGenerateSalt(CrypterOptions).)

Top

### **See Also**

Reference

ShaCrypter Class

CryptSharp Namespace

# ShaCrypterGenerateSalt Method (CrypterOptions)

CryptSharp

Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form algorithm+salt+hash, so the salt is always carried along with the crypted password).

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

### **Parameters**

options

Type: CryptSharpCrypterOptions

Options modifying the salt generation.

### **Return Value**

Type: String
The salt string.

### See Also

#### Reference

**ShaCrypter Class** 

GenerateSalt Overload

CryptSharp Namespace

# **ShaCrypter Properties**

CryptSharp

The ShaCrypter type exposes the following members.

# **Properties**

Name Description		Description
		Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys. (Overrides CrypterProperties.)

Top

### See Also

Reference

ShaCrypter Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com

### **ShaCrypterProperties Property**

CryptSharp

Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys.

Namespace: CryptSharp

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**



### **TraditionalDesCrypter Class**

CryptSharp

Traditional DES crypt.

### **Inheritance Hierarchy**

SystemObject CryptSharpCrypter

Crypt Sharp Traditional Des Crypter

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**



The **TraditionalDesCrypter** type exposes the following members.

### **Constructors**

	Name	Description
<b>≡</b>	TraditionalDesCrypter	Initializes a new instance of the <b>TraditionalDesCrypter</b> class

Top

	Name	Description
=0	CanCrypt	Checks if the particular crypt algorithm is compatible with the salt string or crypted password.  (Overrides CrypterCanCrypt(String).)
=0	Crypt(Byte)	Creates a one-way password hash (crypted password) from password bytes.  (Inherited from Crypter.)
<b>≡</b>	Crypt(String)	Creates a one-way password hash (crypted password) from a password string.  (Inherited from Crypter.)
=0	Crypt(Byte, CrypterOptions)	Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation. (Inherited from Crypter.)

<b>≟</b>	Crypt(String, CrypterOptions)	Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)
∉•	Crypt(String, String)	Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Inherited from Crypter.)
₫•	Crypt(Byte, String)	Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords.  (Overrides CrypterCrypt(Byte, String).)
<b>≓</b>	GenerateSalt	Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password (Inherited from Crypter.)
<b>∃</b>	GenerateSalt(Int32)	Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password.  (Inherited from Crypter.)
<b>∄</b>	GenerateSalt(CrypterOptions)	Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form algorithm+salt+hash, so the salt is always carried along with the crypted password). (Overrides CrypterGenerateSalt(CrypterOptions).)

# **Properties**

Name	Description
Properties	Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys. (Overrides CrypterProperties.)

Top

### See Also

Reference

CryptSharp Namespace

# TraditionalDesCrypter Constructor

CryptSharp

Initializes a new instance of the TraditionalDesCrypter class

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

```
C# VB C++ F#

Copy

public TraditionalDesCrypter()
```

### See Also

Reference

TraditionalDesCrypter Class

CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com

# TraditionalDesCrypter Methods

The TraditionalDesCrypter type exposes the following members.

	Name	Description
<b>≡</b>	CanCrypt	Checks if the particular crypt algorithm is compatible with the salt string or crypted password.  (Overrides CrypterCanCrypt(String).)
<b>≡</b>	Crypt(Byte)	Creates a one-way password hash (crypted password) from password bytes. (Inherited from Crypter.)
<b>≡</b>	Crypt(String)	Creates a one-way password hash (crypted password) from a password string. (Inherited from Crypter.)
<b>≡</b>	Crypt(Byte, CrypterOptions)	Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation. (Inherited from Crypter.)
<b>≡</b>	Crypt(String, CrypterOptions)	Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)
<b>≡</b> ♠	Crypt(String, String)	Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Inherited from Crypter.)
#₩	Crypt(Byte, String)	Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Overrides CrypterCrypt(Byte, String).)

=	•	GenerateSalt	Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)
-	•	GenerateSalt(Int32)	Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password.  (Inherited from Crypter.)
=	•	GenerateSalt(CrypterOptions)	Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form algorithm+salt+hash, so the salt is always carried along with the crypted password). (Overrides CrypterGenerateSalt(CrypterOptions).)

### **See Also**

Reference

TraditionalDesCrypter Class CryptSharp Namespace

### TraditionalDesCrypterCanCrypt Method CryptSharp

Checks if the particular crypt algorithm is compatible with the salt string or crypted password.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

```
public override bool CanCrypt(
    string salt
)

Parameters
salt
    Type: SystemString
    The salt string or crypted password.

Return Value
Type: Boolean
true if the algorithm is compatible.

See Also
Reference
TraditionalDesCrypter Class
```

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com

CryptSharp Namespace

### **Overload List**

	Name	Description
<b>≡</b>	Crypt(Byte)	Creates a one-way password hash (crypted password) from password bytes. (Inherited from Crypter.)
<b>≡</b>	Crypt(String)	Creates a one-way password hash (crypted password) from a password string. (Inherited from Crypter.)
<b>≡</b>	Crypt(Byte, CrypterOptions)	Creates a one-way password hash (crypted password) from password bytes. Options modify the crypt operation. (Inherited from Crypter.)
<b>≡</b>	Crypt(String, CrypterOptions)	Creates a one-way password hash (crypted password) from a password string. Options modify the crypt operation. (Inherited from Crypter.)
≡ <b>◊</b>	Crypt(String, String)	Creates a one-way password hash (crypted password) from a password string and a salt string. The salt can be produced using GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Inherited from Crypter.)
ΞQ	Crypt(Byte, String)	Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using <a href="GenerateSalt(CrypterOptions">GenerateSalt(CrypterOptions</a> ). Because crypted passwords take the form <a href="algorithm+salt+hash">algorithm+salt+hash</a> , if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate *and* verify crypted passwords. (Overrides CrypterCrypt(Byte, String).)

Top

## **See Also**

### Reference

TraditionalDesCrypter Class CryptSharp Namespace

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com

# TraditionalDesCrypterCrypt Method (Byte, CryptSharp String)

Creates a one-way password hash (crypted password) from password bytes and a salt string. The salt can be produced using

GenerateSalt(CrypterOptions). Because crypted passwords take the form algorithm+salt+hash, if you pass a crypted password as the salt parameter, the same algorithm and salt will be used to re-crypt the password. Since randomness comes from the salt, the same salt means the same hash, and so the same crypted password will result. Therefore, this method can both generate \*and\* verify crypted passwords.

Namespace: CryptSharp

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

```
C# VB C++ F#
                                                       Copy
public override string Crypt(
          byte[] password,
          string salt
 )
Parameters
```

password

Type: SystemByte

The bytes of the password.

salt

Type: SystemString

The salt string or crypted password containing a salt string.

#### **Return Value**

Type: String

The crypted password.

### See Also

### Reference

TraditionalDesCrypter Class

**Crypt Overload** 

# CryptSharp Namespace

# $Traditional Des Crypter Generate Salt\ Method {\tt CryptSharp}$

# **Overload List**

	Name	Description
=	GenerateSalt	Generates a salt string with default options. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. (Inherited from Crypter.)
≡•	GenerateSalt(Int32)	Generates a salt string using the specified number of rounds. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password.  (Inherited from Crypter.)
₫ 🕸	GenerateSalt(CrypterOptions)	Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form algorithm+salt+hash, so the salt is always carried along with the crypted password). (Overrides CrypterGenerateSalt(CrypterOptions).)

Top

# **See Also**

Reference

TraditionalDesCrypter Class CryptSharp Namespace

# TraditionalDesCrypterGenerateSalt Method CryptSharp (CrypterOptions)

Generates a salt string. Options are used to modify the salt generation. The purpose of salt is to make dictionary attacks against a whole password database much harder, by causing the crypted password to be different even if two users have the same uncrypted password. Randomness in a crypted password comes from its salt string, as do all recorded options. The same salt string, when combined with the same password, will generate the same crypted password. If the salt string differs, the same password will generate a different crypted password (crypted passwords have the form algorithm+salt+hash, so the salt is always carried along with the crypted password).

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## **Syntax**

## Parameters

options

Type: CryptSharpCrypterOptions

Options modifying the salt generation.

#### **Return Value**

Type: String
The salt string.

### See Also

#### Reference

TraditionalDesCrypter Class

GenerateSalt Overload

CryptSharp Namespace

# **TraditionalDesCrypter Properties**

CryptSharp

The TraditionalDesCrypter type exposes the following members.

# **Properties**

Name	Description
	Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys. (Overrides CrypterProperties.)

Top

# **See Also**

Reference

TraditionalDesCrypter Class CryptSharp Namespace

# TraditionalDesCrypterProperties Property CryptSharp

Properties inherent to the particular crypt algorithm. These cannot be modified. See CrypterProperty for possible keys.

Namespace: CryptSharp

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## **Syntax**



These classes are not directly related to the crypt functionality, but most were necessary for its implementation, so I see no reason not to make them available. You may find them useful for other purposes.

## **Classes**

	Class	Description
43	Base16Encoding	Base-16 binary-to-text encodings.
<b>4</b>	Base2Encoding	Base-2 binary-to-text encodings.
<del>?</del> \$	Base32Encoding	Base-32 binary to text encodings. I needed multiple variations of base-64 for the various crypt algorithms, and base-16 (hex) for test vectors, so base-32 is mostly a freebie. It's great for e-mail verifications, product keys - really anywhere you need someone to type in a randomly-generated code.
<b>4</b>	Base64Encoding	Base-64 binary-to-text encodings.
<b>4</b> \$	BaseEncoding	Performs generic binary-to-text encoding.
<b>4</b> \$	BlowfishCipher	Performs low-level encryption and decryption using the Blowfish cipher.
<b>4</b>	DesCipher	Performs low-level encryption and decryption using the DES cipher.
<b>₩</b>	Pbkdf2	Implements the PBKDF2 key derivation function.
<b>4</b> \$	Salsa20Core	Implements the Salsa20 hash function.
4	SCrypt	Implements the SCrypt key derivation function.
<b>4</b> \$	SecureComparison	Provides comparison methods resistant to timing attack.

# **Delegates**

Delegate	Description
	A callback to map arbitrary characters onto the characters that can be decoded.

# **Enumerations**

Enumeration	Description
EksBlowfishKeyExpansionFlags	Modifiers for Expensive Key Schedule (EKS) Blowfish

LZOTZ	expansion	ha	harmar
N H V	exhansion	11	navioi.

# **Base16Encoding Class**

CryptSharp

Base-16 binary-to-text encodings.

# **Inheritance Hierarchy**

SystemObject CryptSharp.UtilityBase16Encoding

Namespace: CryptSharp.Utility

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**

The **Base16Encoding** type exposes the following members.

# **Properties**

	Name	Description
<b></b> S	Hex	Hexadecimal base-16 uses the numbers 0-9 for 0-9, and the letters A-F for 10-15.

Top

### **See Also**

Reference

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

# **Base16Encoding Properties**

CryptSharp

The Base16Encoding type exposes the following members.

# **Properties**

	Name	Description
s s	Hex	Hexadecimal base-16 uses the numbers 0-9 for 0-9, and the letters A-F for 10-15.

Top

# **See Also**

Reference

**Base16Encoding Class** 

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

# **Base16EncodingHex Property**

CryptSharp

Hexadecimal base-16 uses the numbers 0-9 for 0-9, and the letters A-F for 10-15.

Namespace: CryptSharp.Utility

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## **Syntax**

```
C# VB C++ F#

public static BaseEncoding Hex { get; }

Property Value
Type: BaseEncoding

See Also

Reference
Base16Encoding Class
CryptSharp.Utility Namespace
```

# **Base2Encoding Class**

CryptSharp

Base-2 binary-to-text encodings.

# **Inheritance Hierarchy**

SystemObject CryptSharp.UtilityBase2Encoding

Namespace: CryptSharp.Utility

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**

```
C# VB C++ F#

Copy

public static class Base2Encoding
```

The **Base2Encoding** type exposes the following members.

# **Properties**

	Name	Description
<b>₽</b> S	Binary	Binary. Useful for debugging.

Top

### See Also

Reference

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

# **Base2Encoding Properties**

CryptSharp

The Base2Encoding type exposes the following members.

# **Properties**

	Name	Description
s s	Binary	Binary. Useful for debugging.

Top

# **See Also**

#### Reference

Base2Encoding Class

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

# **Base2EncodingBinary Property**

CryptSharp

Binary. Useful for debugging.

Namespace: CryptSharp.Utility

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**

```
C# VB C++ F#
                                                        Copy
 public static BaseEncoding Binary { get; }
Property Value
Type: BaseEncoding
See Also
```

Reference

Base2Encoding Class

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

# **Base32Encoding Class**

CryptSharp

Base-32 binary to text encodings. I needed multiple variations of base-64 for the various crypt algorithms, and base-16 (hex) for test vectors, so base-32 is mostly a freebie. It's great for e-mail verifications, product keys - really anywhere you need someone to type in a randomly-generated code.

# **Inheritance Hierarchy**

SystemObject CryptSharp.UtilityBase32Encoding

Namespace: CryptSharp.Utility

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**



The **Base32Encoding** type exposes the following members.

# **Properties**

	Name	Description
S	Crockford	Crockford base-32 is somewhat traditional, but still better than the RFC 4648 standard. It is specified at http://www.crockford.com/wrmg/base32.html.
S	ZBase32	z-base-32 is a lowercase base-32 encoding designed to be easily hand-written and read. It is specified at http://www.zer7.com/files/oss/cryptsharp/zbase32.txt.

Top

## See Also

Reference

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

# **Base32Encoding Properties**

CryptSharp

The Base32Encoding type exposes the following members.

# **Properties**

	Name	Description
≝°S	Crockford	Crockford base-32 is somewhat traditional, but still better than the RFC 4648 standard. It is specified at http://www.crockford.com/wrmg/base32.html.
<b>₽</b> S	ZBase32	z-base-32 is a lowercase base-32 encoding designed to be easily hand-written and read. It is specified at http://www.zer7.com/files/oss/cryptsharp/zbase32.txt.

#### Top

# See Also

#### Reference

Base32Encoding Class

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

# **Base32EncodingCrockford Property**

CryptSharp

Crockford base-32 is somewhat traditional, but still better than the RFC 4648 standard. It is specified at http://www.crockford.com/wrmg/base32.html.

Namespace: CryptSharp.Utility

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## **Syntax**

```
C# VB C++ F#

public static BaseEncoding Crockford { get; }

Property Value
Type: BaseEncoding

See Also

Reference
Base32Encoding Class
CryptSharp.Utility Namespace
```

# Base32EncodingZBase32 Property

CryptSharp

z-base-32 is a lowercase base-32 encoding designed to be easily hand-written and read. It is specified at

http://www.zer7.com/files/oss/cryptsharp/zbase32.txt.

Namespace: CryptSharp.Utility

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**

```
C# VB C++ F#
                                                        Copy
public static BaseEncoding ZBase32 { get; }
Property Value
```

Type: BaseEncoding

## See Also

Reference

Base32Encoding Class

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

# **Base64Encoding Class**

CryptSharp

Base-64 binary-to-text encodings.

# **Inheritance Hierarchy**

SystemObject CryptSharp.UtilityBase64Encoding

Namespace: CryptSharp.Utility

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**

```
C# VB C++ F#

public static class Base64Encoding
```

The **Base64Encoding** type exposes the following members.

# **Properties**

	Name	Description	
S	Blowfish Blowfish crypt orders characters differently from standard crypt, and begins encoding from the most-significant bit instead of the least-significant bit.		
<b>≅</b> S	UnixCrypt	Traditional DES crypt base-64, as seen on Unix /etc/passwd, many websites, database servers, etc.	
≝°S	UnixMD5	MD5, SHA256, and SHA512 crypt base-64, as seen on Unix /etc/passwd, many websites, database servers, etc.	

Top

## **See Also**

Reference

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

# **Base64Encoding Properties**

CryptSharp

The Base64Encoding type exposes the following members.

# **Properties**

	Name	Description	
<b>≅</b> S	Blowfish	Blowfish crypt orders characters differently from standard crypt, and begins encoding from the most-significant bit instead of the least-significant bit.	
<b>≅</b> S	UnixCrypt	Traditional DES crypt base-64, as seen on Unix /etc/passwd, many websites, database servers, etc.	
≝°S	UnixMD5	MD5, SHA256, and SHA512 crypt base-64, as seen on Unix /etc/passwd, many websites, database servers, etc.	

### Top

# See Also

#### Reference

Base64Encoding Class

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

# **Base64EncodingBlowfish Property**

CryptSharp

Blowfish crypt orders characters differently from standard crypt, and begins encoding from the most-significant bit instead of the least-significant bit.

Namespace: CryptSharp.Utility

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## **Syntax**

```
C# VB C++ F#

public static BaseEncoding Blowfish { get; }

Property Value
Type: BaseEncoding

See Also

Reference
Base64Encoding Class
CryptSharp.Utility Namespace
```

# **Base64EncodingUnixCrypt Property**

CryptSharp

Traditional DES crypt base-64, as seen on Unix /etc/passwd, many websites, database servers, etc.

Namespace: CryptSharp.Utility

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## **Syntax**

```
C# VB C++ F#

public static BaseEncoding UnixCrypt { get; }

Property Value
Type: BaseEncoding

See Also

Reference
```

Base64Encoding Class

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

# Base64EncodingUnixMD5 Property

CryptSharp

MD5, SHA256, and SHA512 crypt base-64, as seen on Unix /etc/passwd, many websites, database servers, etc.

Namespace: CryptSharp.Utility

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## **Syntax**

```
C# VB C++ F#

public static BaseEncoding UnixMD5 { get; }

Property Value
Type: BaseEncoding

See Also

Reference
Base64Encoding Class
CryptSharp.Utility Namespace
```

# **BaseEncoding Class**

CryptSharp

Performs generic binary-to-text encoding.

# **Inheritance Hierarchy**

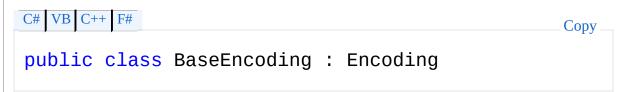
SystemObject System.TextEncoding

CryptSharp.UtilityBaseEncoding

Namespace: CryptSharp.Utility

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## **Syntax**



The **BaseEncoding** type exposes the following members.

#### **Constructors**

	Name	Description
<b>≡</b>	BaseEncoding(String, Boolean)	Defines a binary-to-text encoding.
<b>≟</b>	BaseEncoding(String, Boolean, IDictionaryChar, Int32, BaseEncodingDecodeFilterCallback)	Defines a binary-to-text encoding. Additional decode characters let you add aliases, and a filter callback can be used to make decoding case-insensitive among other things.

#### Top

# **Methods**

	Name	Description
<b>≡</b>	GetByteCount	When overridden in a derived class, calculates the number of bytes produced by encoding a set of characters from the specified character array.  (Overrides EncodingGetByteCount(Char, Int32, Int32).)
<b>≡</b>	GetBytes(Char, Int32, Int32, Byte, Int32)	When overridden in a derived class, encodes a set of characters from the specified character array into the specified byte array. (Overrides EncodingGetBytes(Char, Int32, Int32, Byte, Int32).)
<b>≡</b>	GetBytes(Char, Int32,	Converts characters from their text representation to a binary

	Int32, Byte, Int32, Int32)	representation.
<b>≡</b>	GetChar	Gets the character corresponding to the specified value.
<b>≓</b>	GetCharCount	When overridden in a derived class, calculates the number of characters produced by decoding a sequence of bytes from the specified byte array.  (Overrides EncodingGetCharCount(Byte, Int32, Int32).)
<b>≡</b>	GetChars(Byte, Int32, Int32, Char, Int32)	When overridden in a derived class, decodes a sequence of bytes from the specified byte array into the specified character array. (Overrides EncodingGetChars(Byte, Int32, Int32, Char, Int32).)
<b>≡</b>	GetChars(Byte, Int32, Int32, Char, Int32, Int32)	Converts bytes from their binary representation to a text representation.
<b>≡</b>	GetMaxByteCount	When overridden in a derived class, calculates the maximum number of bytes produced by encoding the specified number of characters. (Overrides EncodingGetMaxByteCount(Int32).)
<b>=</b>	GetMaxCharCount	When overridden in a derived class, calculates the maximum number of characters produced by decoding the specified number of bytes. (Overrides EncodingGetMaxCharCount(Int32).)
<b>≡</b>	GetValue	Gets the value corresponding to the specified character.

## Top

# **Properties**

Name	Description
BitMask	The bit mask for a single character in the current encoding.
BitsPerCharacter	The number of bits per character in the current encoding.
MsbComesFirst	true if the encoding begins with the most-significant bit of each byte. Otherwise, the encoding begins with the least-significant bit.

## Top

# **See Also**

#### Reference

CryptSharp.Utility Namespace

# **BaseEncoding Constructor**

CryptSharp

# **Overload List**

	Name	Description
<b>≡</b>	BaseEncoding(String, Boolean)	Defines a binary-to-text encoding.
=0	BaseEncoding(String, Boolean, IDictionaryChar, Int32, BaseEncodingDecodeFilterCallback)	Defines a binary-to-text encoding. Additional decode characters let you add aliases, and a filter callback can be used to make decoding case-insensitive among other things.

Top

# **See Also**

Reference

BaseEncoding Class

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

# BaseEncoding Constructor (String, Boolean) CryptSharp

Defines a binary-to-text encoding.

Namespace: CryptSharp.Utility

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**

```
public BaseEncoding(
          string characterSet,
          bool msbComesFirst
)
```

#### **Parameters**

characterSet

Type: SystemString

The characters of the encoding.

*msbComesFirst* 

Type: SystemBoolean

true to begin with the most-significant bit of each byte. Otherwise, the encoding begins with the least-significant bit.

## **See Also**

#### Reference

BaseEncoding Class

BaseEncoding Overload

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

### BaseEncoding Constructor (String, Boolean, IDictionaryChar, Int32, CryptSharp BaseEncodingDecodeFilterCallback)

Defines a binary-to-text encoding. Additional decode characters let you add aliases, and a filter callback can be used to make decoding case-insensitive among other things.

Namespace: CryptSharp.Utility

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## **Syntax**

```
C# VB C++ F#
                                                               Copy
 public BaseEncoding(
           string characterSet,
           bool msbComesFirst,
           IDictionary<char, int> additionalDecodeChara
           BaseEncodingDecodeFilterCallback decodeFilte
 )
Parameters
characterSet
  Type: SystemString
  The characters of the encoding.
msbComesFirst
  Type: SystemBoolean
  true to begin with the most-significant bit of each byte. Otherwise, the
  encoding begins with the least-significant bit.
additionalDecodeCharacters
  Type: System.Collections.GenericIDictionaryChar, Int32
  A dictionary of alias characters, or null if no aliases are desired.
decodeFilterCallback
```

Type: CryptSharp.UtilityBaseEncodingDecodeFilterCallback

A callback to map arbitrary characters onto the characters that can be decoded.

# See Also

#### Reference

BaseEncoding Class BaseEncoding Overload CryptSharp.Utility Namespace

The BaseEncoding type exposes the following members.

# **Methods**

	Name	Description
<b>≟</b>	GetByteCount	When overridden in a derived class, calculates the number of bytes produced by encoding a set of characters from the specified character array.  (Overrides EncodingGetByteCount(Char, Int32, Int32).)
<b>≡</b>	GetBytes(Char, Int32, Int32, Byte, Int32)	When overridden in a derived class, encodes a set of characters from the specified character array into the specified byte array. (Overrides EncodingGetBytes(Char, Int32, Int32, Byte, Int32).)
<b>=</b>	GetBytes(Char, Int32, Int32, Byte, Int32, Int32)	Converts characters from their text representation to a binary representation.
<b>≡</b>	GetChar	Gets the character corresponding to the specified value.
=0	GetCharCount	When overridden in a derived class, calculates the number of characters produced by decoding a sequence of bytes from the specified byte array.  (Overrides EncodingGetCharCount(Byte, Int32, Int32).)
<b>≡</b>	GetChars(Byte, Int32, Int32, Char, Int32)	When overridden in a derived class, decodes a sequence of bytes from the specified byte array into the specified character array. (Overrides EncodingGetChars(Byte, Int32, Int32, Char, Int32).)
<b>≡</b>	GetChars(Byte, Int32, Int32, Char, Int32, Int32)	Converts bytes from their binary representation to a text representation.
<b>≡</b>	GetMaxByteCount	When overridden in a derived class, calculates the maximum number of bytes produced by encoding the specified number of characters. (Overrides EncodingGetMaxByteCount(Int32).)
<b>=</b>	GetMaxCharCount	When overridden in a derived class, calculates the maximum number of characters produced by decoding the specified number of bytes. (Overrides EncodingGetMaxCharCount(Int32).)
<b>≡</b>	GetValue	Gets the value corresponding to the specified character.

Top

# See Also

Reference

BaseEncoding Class

# CryptSharp.Utility Namespace

# **BaseEncodingGetByteCount Method**

CryptSharp

When overridden in a derived class, calculates the number of bytes produced by encoding a set of characters from the specified character array.

Namespace: CryptSharp.Utility

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## **Syntax**

```
public override int GetByteCount(
          char[] chars,
          int index,
          int count
)
```

#### **Parameters**

chars

Type: SystemChar

The character array containing the set of characters to encode.

index

Type: SystemInt32

The index of the first character to encode.

count

Type: SystemInt32

The number of characters to encode.

## Return Value

Type: Int32

The number of bytes produced by encoding the specified characters.

# **Exceptions**

Exception	Condition
ArgumentNullException	chars is null.
ArgumentOutOfRangeException	<i>index</i> or <i>count</i> is less than zeroor- <i>index</i> and <i>count</i> do not denote a valid range in <i>chars</i> .
EncoderFallbackException	A fallback occurred (see Understanding Encodings for complete

 $explanation) \hbox{-} and \hbox{-} Encoder Fallback is set to} \\ Encoder Exception Fallback.$ 

# See Also

#### Reference

BaseEncoding Class CryptSharp.Utility Namespace

# BaseEncodingGetBytes Method

CryptSharp

# **Overload List**

	Name	Description
<b>=</b>	GetBytes(Char, Int32, Int32, Byte, Int32)	When overridden in a derived class, encodes a set of characters from the specified character array into the specified byte array. (Overrides EncodingGetBytes(Char, Int32, Int32, Byte, Int32).)
=0	GetBytes(Char, Int32, Int32, Byte, Int32, Int32)	Converts characters from their text representation to a binary representation.

Top

# **See Also**

Reference

BaseEncoding Class

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

# BaseEncodingGetBytes Method (Char, **Int32**, **Int32**, **Byte**, **Int32**)

CryptSharp

When overridden in a derived class, encodes a set of characters from the specified character array into the specified byte array.

Namespace: CryptSharp.Utility

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

```
Syntax
```

```
C# VB C++ F#
                                                                  Copy
 public override int GetBytes(
            char[] chars,
            int charIndex,
            int charCount,
            byte[] bytes,
            int byteIndex
Parameters
chars
  Type: SystemChar
  The character array containing the set of characters to encode.
charIndex
  Type: SystemInt32
  The index of the first character to encode.
charCount
  Type: SystemInt32
  The number of characters to encode.
bytes
  Type: SystemByte
  The byte array to contain the resulting sequence of bytes.
byteIndex |
  Type: SystemInt32
  The index at which to start writing the resulting sequence of bytes.
```

**Return Value** Type: Int32 The actual number of bytes written into *bytes*.

# **Exceptions**

Exception	Condition
ArgumentNullException	chars is nullor- bytes is null.
ArgumentOutOfRangeException	charIndex or charCount or byteIndex is less than zeroor- charIndex and charCount do not denote a valid range in chars or- byteIndex is not a valid index in bytes.
ArgumentException	<i>bytes</i> does not have enough capacity from <i>byteIndex</i> to the end of the array to accommodate the resulting bytes.
EncoderFallbackException	A fallback occurred (see Understanding Encodings for complete explanation)-and-EncoderFallback is set to EncoderExceptionFallback.

# See Also

#### Reference

BaseEncoding Class
GetBytes Overload
CryptSharp.Utility Namespace

# BaseEncodingGetBytes Method (Char, Int32, Int32, Byte, Int32, Int32)

CryptSharp

Converts characters from their text representation to a binary representation.

Namespace: CryptSharp.Utility

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## **Syntax**

```
C# VB C++ F#
                                                                Copy
 public int GetBytes(
            char[] chars,
            int charIndex,
            int charCount,
            byte[] bytes,
            int byteIndex,
            int byteCount
Parameters
chars
  Type: SystemChar
  An input array of characters.
charIndex
  Type: SystemInt32
  The index of the first character.
charCount
  Type: SystemInt32
  The number of characters to read.
bytes
  Type: SystemByte
  An output array of bytes.
byteIndex |
  Type: SystemInt32
  The index of the first byte.
byteCount
  Type: SystemInt32
```

The number of bytes to write.

**Return Value** 

Type: Int32

The number of bytes written.

#### **See Also**

Reference

BaseEncoding Class

GetBytes Overload

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

## BaseEncodingGetChar Method

CryptSharp

Gets the character corresponding to the specified value.

Namespace: CryptSharp.Utility

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

```
Syntax
```

Copyright 2010-2014 James F. Bellinger

# BaseEncodingGetCharCount Method

CryptSharp

When overridden in a derived class, calculates the number of characters produced by decoding a sequence of bytes from the specified byte array.

Namespace: CryptSharp.Utility

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

#### **Syntax**

#### **Parameters**

bytes

Type: SystemByte

The byte array containing the sequence of bytes to decode.

index

Type: SystemInt32

The index of the first byte to decode.

count

Type: SystemInt32

The number of bytes to decode.

#### **Return Value**

Type: Int32

The number of characters produced by decoding the specified sequence of bytes.

# **Exceptions**

Exception	Condition
ArgumentNullException	bytes is null.
ArgumentOutOfRangeException	<i>index</i> or <i>count</i> is less than zeroor- <i>index</i> and <i>count</i> do not denote a valid range in <i>bytes</i> .

Γ	DecoderFallbackException	A fallback occurred (see Understanding Encodings for complete explanation)-and-DecoderFallback is set to DecoderExceptionFallback.
		1 /

# See Also

#### Reference

BaseEncoding Class CryptSharp.Utility Namespace

# ${\bf Base Encoding Get Chars\ Method}$

CryptSharp

## **Overload List**

	Name	Description	
<b>≡</b>	GetChars(Byte, Int32, Int32, Char, Int32)	When overridden in a derived class, decodes a sequence of bytes from the specified byte array into the specified character array. (Overrides EncodingGetChars(Byte, Int32, Int32, Char, Int32).)	
<b>≡</b>	GetChars(Byte, Int32, Int32, Char, Int32, Int32)	Converts bytes from their binary representation to a text representation.	

Top

#### **See Also**

Reference

BaseEncoding Class

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

# BaseEncodingGetChars Method (Byte, **Int32**, **Int32**, **Char**, **Int32**)

CryptSharp

When overridden in a derived class, decodes a sequence of bytes from the specified byte array into the specified character array.

Namespace: CryptSharp.Utility

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

```
Syntax
```

```
C# VB C++ F#
                                                                 Copy
 public override int GetChars(
            byte[] bytes,
            int byteIndex,
            int byteCount,
            char[] chars,
            int charIndex
Parameters
bytes
  Type: SystemByte
  The byte array containing the sequence of bytes to decode.
byteIndex |
  Type: SystemInt32
  The index of the first byte to decode.
byteCount
  Type: SystemInt32
  The number of bytes to decode.
chars
  Type: SystemChar
  The character array to contain the resulting set of characters.
charIndex
  Type: SystemInt32
```

The index at which to start writing the resulting set of characters.

**Return Value** Type: Int32 The actual number of characters written into *chars*.

# **Exceptions**

Exception	Condition
ArgumentNullException	bytes is nullor- chars is null.
ArgumentOutOfRangeException	byteIndex or byteCount or charIndex is less than zeroor-byteindex and byteCount do not denote a valid range in bytesor-charIndex is not a valid index in chars.
ArgumentException	<i>chars</i> does not have enough capacity from <i>charIndex</i> to the end of the array to accommodate the resulting characters.
DecoderFallbackException	A fallback occurred (see Understanding Encodings for complete explanation)-and-DecoderFallback is set to DecoderExceptionFallback.

## See Also

#### Reference

BaseEncoding Class

GetChars Overload

CryptSharp.Utility Namespace

## BaseEncodingGetChars Method (Byte, Int32, Int32, Char, Int32, Int32)

CryptSharp

Converts bytes from their binary representation to a text representation.

Namespace: CryptSharp.Utility

Type: SystemInt32

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

#### **Syntax**

```
C# VB C++ F#
                                                                Copy
 public int GetChars(
            byte[] bytes,
            int byteIndex,
            int byteCount,
            char[] chars,
            int charIndex,
            int charCount
Parameters
bytes
  Type: SystemByte
  An input array of bytes.
byteIndex |
  Type: SystemInt32
  The index of the first byte.
byteCount
  Type: SystemInt32
  The number of bytes to read.
chars
  Type: SystemChar
  An output array of characters.
charIndex
  Type: SystemInt32
  The index of the first character.
charCount
```

The number of characters to write.

#### **Return Value**

Type: Int32

The number of characters written.

#### See Also

#### Reference

BaseEncoding Class

GetChars Overload

CryptSharp.Utility Namespace

### BaseEncodingGetMaxByteCount Method CryptSharp

When overridden in a derived class, calculates the maximum number of bytes produced by encoding the specified number of characters.

Namespace: CryptSharp.Utility

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

#### **Syntax**

```
C# VB C++ F#

public override int GetMaxByteCount(
    int charCount
)
```

#### **Parameters**

charCount

Type: SystemInt32

The number of characters to encode.

# Return Value Type: Int32

The maximum number of bytes produced by encoding the specified number of characters.

#### **Exceptions**

Exception	Condition
ArgumentOutOfRangeException	charCount is less than zero.
EncoderFallbackException	A fallback occurred (see Understanding Encodings for complete explanation)-and-EncoderFallback is set to EncoderExceptionFallback.

## See Also

#### Reference

BaseEncoding Class

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

## BaseEncodingGetMaxCharCount Method CryptSharp

When overridden in a derived class, calculates the maximum number of characters produced by decoding the specified number of bytes.

Namespace: CryptSharp.Utility

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

#### **Syntax**

```
C# VB C++ F#

public override int GetMaxCharCount(
    int byteCount
)
```

#### **Parameters**

*byteCount* 

Type: SystemInt32

The number of bytes to decode.

# Return Value Type: Int32

The maximum number of characters produced by decoding the specified number of bytes.

#### **Exceptions**

Exception	Condition
ArgumentOutOfRangeException	byteCount is less than zero.
DecoderFallbackException	A fallback occurred (see Understanding Encodings for complete explanation)-and-DecoderFallback is set to DecoderExceptionFallback.

#### See Also

#### Reference

BaseEncoding Class

CryptSharp.Utility Namespace

#### Copyright 2010-2014 James F. Bellinger

## **BaseEncodingGetValue Method**

CryptSharp

Gets the value corresponding to the specified character.

Namespace: CryptSharp.Utility

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

#### **Syntax**

#### **Parameters**

character

Type: SystemChar

A character.

#### **Return Value**

Type: Int32

A value, or -1 if the character is not part of the encoding.

#### See Also

Reference

BaseEncoding Class

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

# **BaseEncoding Properties**

CryptSharp

The BaseEncoding type exposes the following members.

# **Properties**

Name	Description	
BitMask	The bit mask for a single character in the current encoding.	
BitsPerCharacter	The number of bits per character in the current encoding.	
MsbComesFirst true if the encoding begins with the most-significant bit of each byt Otherwise, the encoding begins with the least-significant bit.		

#### Top

## **See Also**

#### Reference

BaseEncoding Class

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

## **BaseEncodingBitMask Property**

CryptSharp

The bit mask for a single character in the current encoding.

Namespace: CryptSharp.Utility

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## **Syntax**

```
C# VB C++ F#

Copy

public int BitMask { get; }

Property Value
Type: Int32

See Also

Reference
BaseEncoding Class
CryptSharp.Utility Namespace
```

Copyright 2010-2014 James F. Bellinger

## BaseEncodingBitsPerCharacter Property CryptSharp

The number of bits per character in the current encoding.

Namespace: CryptSharp.Utility

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

```
C# VB C++ F# Copy

public int BitsPerCharacter { get; }

Property Value
Type: Int32

See Also

Reference
BaseEncoding Class
CryptSharp.Utility Namespace
```

Copyright 2010-2014 James F. Bellinger

## **BaseEncodingMsbComesFirst Property**

CryptSharp

true if the encoding begins with the most-significant bit of each byte. Otherwise, the encoding begins with the least-significant bit.

Namespace: CryptSharp.Utility

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

```
C# VB C++ F#

public bool MsbComesFirst { get; }

Property Value
Type: Boolean

See Also

Reference
BaseEncoding Class
CryptSharp.Utility Namespace
```

# BaseEncodingDecodeFilterCallback Delegate

CryptSharp

A callback to map arbitrary characters onto the characters that can be decoded.

Namespace: CryptSharp.Utility

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

#### **Syntax**

#### **Parameters**

originalCharacter

Type: SystemChar The original character.

#### **Return Value**

Type: Char

the replacement character.

#### **See Also**

Reference

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

## **BlowfishCipher Class**

CryptSharp

Performs low-level encryption and decryption using the Blowfish cipher.

#### **Inheritance Hierarchy**

SystemObject CryptSharp.UtilityBlowfishCipher

Namespace: CryptSharp.Utility

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

#### **Syntax**

```
C# VB C++ F#

Copy

public class BlowfishCipher : IDisposable
```

The **BlowfishCipher** type exposes the following members.

#### **Methods**

	Name	Description
<b>=</b>	BCrypt	Uses the cipher to generate a BCrypt hash.
<b>=\$S</b>	BCrypt(Byte, Byte, Int32)	Uses the given key, salt, and cost to generate a BCrypt hash.
<b>=\$S</b>	BCrypt(Byte, Byte, Int32, EksBlowfishKeyExpansionFlags)	Uses the given key, salt, and cost to generate a BCrypt hash. Flags may modify the key expansion.
= <b>0</b> S	Create	Creates a Blowfish cipher using the provided key.
= <b>0</b> S	CreateEks(Byte, Byte, Int32)	Performs an Expensive Key Schedule (EKS) Blowfish key expansion and creates a Blowfish cipher using the result.
ĕ∳S	CreateEks(Byte, Byte, Int32, EksBlowfishKeyExpansionFlags)	Performs an Expensive Key Schedule (EKS) Blowfish key expansion and creates a Blowfish cipher using the result. Flags may modify the key expansion.
<b>≡</b>	Decipher(Byte, Int32)	Reverses the encipherment of eight bytes of data inplace.
<b>≡</b>	Decipher(UInt32, UInt32)	Reverses the encipherment of eight bytes of data.
<b>≡</b>	Decipher(Byte, Int32, Byte, Int32)	Reverses the encipherment of eight bytes of data from one buffer and places the result in another buffer.
<b>≡</b>	Dispose	Clears all memory used by the cipher.

<b>=</b>	Encipher(Byte, Int32)	Enciphers eight bytes of data in-place.
<b>=</b>	Encipher(UInt32, UInt32)	Enciphers eight bytes of data.
<b>=</b>	Encipher(Byte, Int32, Byte, Int32)	Enciphers eight bytes of data from one buffer and places the result in another buffer.

#### Top

# **Properties**

	Name	Description	
s s	BCryptLength	The number of bytes returned by BCrypt.	
<b>₽</b> S	BCryptMagic	The special string used encrypted in the BCrypt algorithm.	
	IsKeyWeak	A Blowfish key is weak if one of its S-boxes has a duplicate entry. See http://www.schneier.com/paper-blowfish-oneyear.html for more information.	

#### Top

## See Also

Reference

CryptSharp.Utility Namespace

The BlowfishCipher type exposes the following members.

## **Methods**

	Name	Description
<b>≡</b>	BCrypt	Uses the cipher to generate a BCrypt hash.
∉ <b>≬S</b>	BCrypt(Byte, Byte, Int32)	Uses the given key, salt, and cost to generate a BCrypt hash.
<b>=\$S</b>	BCrypt(Byte, Byte, Int32, EksBlowfishKeyExpansionFlags)	Uses the given key, salt, and cost to generate a BCrypt hash. Flags may modify the key expansion.
<b>=\$S</b>	Create	Creates a Blowfish cipher using the provided key.
ĕ∳S	CreateEks(Byte, Byte, Int32)	Performs an Expensive Key Schedule (EKS) Blowfish key expansion and creates a Blowfish cipher using the result.
<b>=\$S</b>	CreateEks(Byte, Byte, Int32, EksBlowfishKeyExpansionFlags)	Performs an Expensive Key Schedule (EKS) Blowfish key expansion and creates a Blowfish cipher using the result. Flags may modify the key expansion.
<b>≡</b>	Decipher(Byte, Int32)	Reverses the encipherment of eight bytes of data inplace.
<b>=</b>	Decipher(UInt32, UInt32)	Reverses the encipherment of eight bytes of data.
<b>≡</b>	Decipher(Byte, Int32, Byte, Int32)	Reverses the encipherment of eight bytes of data from one buffer and places the result in another buffer.
<b>=</b>	Dispose	Clears all memory used by the cipher.
<b>=</b>	Encipher(Byte, Int32)	Enciphers eight bytes of data in-place.
<b>≡</b>	Encipher(UInt32, UInt32)	Enciphers eight bytes of data.
<b>≡</b>	Encipher(Byte, Int32, Byte, Int32)	Enciphers eight bytes of data from one buffer and places the result in another buffer.

Top

## See Also

Reference

BlowfishCipher Class

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

# **BlowfishCipherBCrypt Method**

CryptSharp

## **Overload List**

	Name	Description
<b>≡</b>	BCrypt	Uses the cipher to generate a BCrypt hash.
∉ <b>≬S</b>	BCrypt(Byte, Byte, Int32)	Uses the given key, salt, and cost to generate a BCrypt hash.
=QS	BCrypt(Byte, Byte, Int32, EksBlowfishKeyExpansionFlags)	Uses the given key, salt, and cost to generate a BCrypt hash. Flags may modify the key expansion.

Top

## See Also

Reference

BlowfishCipher Class

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

# **BlowfishCipherBCrypt Method**

CryptSharp

Uses the cipher to generate a BCrypt hash.

Namespace: CryptSharp.Utility

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## **Syntax**

```
C# VB C++ F#

public byte[] BCrypt()

Return Value
Type: Byte
A BCrypt hash.
```

#### **See Also**

Reference

BlowfishCipher Class

**BCrypt Overload** 

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

# BlowfishCipherBCrypt Method (Byte, Byte, CryptSharp Int32)

Uses the given key, salt, and cost to generate a BCrypt hash.

Namespace: CryptSharp.Utility

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

#### **Syntax**

```
C# VB C++ F#
                                                                  Copy
 public static byte[] BCrypt(
            byte[] key,
            byte[] salt,
            int cost
 )
Parameters
key
  Type: SystemByte
  The key. This must be between 1 and 72 bytes. Unlike BlowfishCrypter,
  this method does NOT automatically add a null byte to the key.
salt
  Type: SystemByte
  The salt. This must be 16 bytes.
cost
  Type: SystemInt32
  The expansion cost. This is a value between 4 and 31, specifying the
  logarithm of the number of iterations.
Return Value
```

Type: Byte A BCrypt hash.

#### **See Also**

#### Reference

BlowfishCipher Class BCrypt Overload

# CryptSharp.Utility Namespace

# BlowfishCipherBCrypt Method (Byte, Byte, CryptSharp Int32, EksBlowfishKeyExpansionFlags)

Uses the given key, salt, and cost to generate a BCrypt hash. Flags may modify the key expansion.

Namespace: CryptSharp.Utility

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

```
Syntax
```

```
C# VB C++ F#
                                                                 Copy
 public static byte[] BCrypt(
            byte[] key,
            byte[] salt,
            int cost,
            EksBlowfishKeyExpansionFlags flags
 )
Parameters
key
  Type: SystemByte
  The key. This must be between 1 and 72 bytes. Unlike BlowfishCrypter,
  this method does NOT automatically add a null byte to the key.
salt
  Type: SystemByte
  The salt. This must be 16 bytes.
cost
  Type: SystemInt32
  The expansion cost. This is a value between 4 and 31, specifying the
  logarithm of the number of iterations.
flags
  Type: CryptSharp.UtilityEksBlowfishKeyExpansionFlags
  Flags modifying the key expansion.
Return Value
Type: Byte
A BCrypt hash.
```

# See Also

#### Reference

BlowfishCipher Class

**BCrypt Overload** 

CryptSharp.Utility Namespace

## **BlowfishCipherCreate Method**

CryptSharp

Creates a Blowfish cipher using the provided key.

Namespace: CryptSharp.Utility

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

#### Parameters

key

Type: SystemByte

The Blowfish key. This must be between 4 and 56 bytes.

#### **Return Value**

Type: BlowfishCipher A Blowfish cipher.

#### See Also

#### Reference

BlowfishCipher Class

CryptSharp.Utility Namespace

#### Copyright 2010-2014 James F. Bellinger

# ${\bf Blow fish Cipher Create Eks\ Method}$

CryptSharp

## **Overload List**

	Name	Description
<b>=♦S</b>	CreateEks(Byte, Byte, Int32)	Performs an Expensive Key Schedule (EKS) Blowfish key expansion and creates a Blowfish cipher using the result.
<b>=\$</b> \$	CreateEks(Byte, Byte, Int32, EksBlowfishKeyExpansionFlags)	Performs an Expensive Key Schedule (EKS) Blowfish key expansion and creates a Blowfish cipher using the result. Flags may modify the key expansion.

Top

## **See Also**

Reference

BlowfishCipher Class

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

# BlowfishCipherCreateEks Method (Byte, Byte, Int32)

CryptSharp

Performs an Expensive Key Schedule (EKS) Blowfish key expansion and creates a Blowfish cipher using the result.

Namespace: CryptSharp.Utility

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

#### **Syntax**

key

Type: SystemByte

The key. This must be between 1 and 72 bytes. Unlike BlowfishCrypter, this method does NOT automatically add a null byte to the key.

salt

Type: SystemByte

The salt. This must be 16 bytes.

cost

Type: SystemInt32

The expansion cost. This is a value between 4 and 31, specifying the logarithm of the number of iterations.

#### **Return Value**

Type: BlowfishCipher A Blowfish cipher.

#### **See Also**

#### Reference

BlowfishCipher Class

CreateEks Overload

# CryptSharp.Utility Namespace

# BlowfishCipherCreateEks Method (Byte, Byte, Int32,

CryptSharp

## EksBlowfishKeyExpansionFlags)

Performs an Expensive Key Schedule (EKS) Blowfish key expansion and creates a Blowfish cipher using the result. Flags may modify the key expansion.

Namespace: CryptSharp.Utility

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

#### **Syntax**

**Return Value** 

```
C# VB C++ F#
                                                                 Copy
 public static BlowfishCipher CreateEks(
            byte[] key,
            byte[] salt,
            int cost,
            EksBlowfishKeyExpansionFlags flags
 )
Parameters
kev
  Type: SystemByte
  The key. This must be between 1 and 72 bytes. Unlike BlowfishCrypter,
  this method does NOT automatically add a null byte to the key.
salt
  Type: SystemByte
  The salt. This must be 16 bytes.
cost
  Type: SystemInt32
  The expansion cost. This is a value between 4 and 31, specifying the
  logarithm of the number of iterations.
flags
  Type: CryptSharp.UtilityEksBlowfishKeyExpansionFlags
  Flags modifying the key expansion.
```

Type: BlowfishCipher A Blowfish cipher.

# See Also

Reference

BlowfishCipher Class CreateEks Overload

CryptSharp.Utility Namespace

# **BlowfishCipherDecipher Method**

CryptSharp

# **Overload List**

		Name	Description
=	•	Decipher(Byte, Int32)	Reverses the encipherment of eight bytes of data in-place.
=	•	Decipher(UInt32, UInt32)	Reverses the encipherment of eight bytes of data.
=(	•	Decipher(Byte, Int32, Byte, Int32)	Reverses the encipherment of eight bytes of data from one buffer and places the result in another buffer.

#### Top

## **See Also**

Reference

BlowfishCipher Class

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

# BlowfishCipherDecipher Method (Byte, Int32)

CryptSharp

Reverses the encipherment of eight bytes of data in-place.

Namespace: CryptSharp.Utility

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

#### **Syntax**

```
C# VB C++ F#
                                                                 Copy
 public void Decipher(
            byte[] buffer,
            int offset
 )
Parameters
buffer
  Type: SystemByte
  The buffer containing the data.
offset
  Type: SystemInt32
  The offset of the first byte to decipher.
See Also
Reference
BlowfishCipher Class
Decipher Overload
```

Copyright 2010-2014 James F. Bellinger

Send comments on this topic to jfb@zer7.com

CryptSharp.Utility Namespace

# BlowfishCipherDecipher Method (UInt32, UInt32)

CryptSharp

Reverses the encipherment of eight bytes of data.

Namespace: CryptSharp.Utility

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

#### **Syntax**

```
C# VB C++ F#
                                                                Copy
 public void Decipher(
            ref uint x1,
            ref uint xr
 )
Parameters
хl
  Type: SystemUInt32
  The first four bytes.
xr
  Type: SystemUInt32
  The last four bytes.
See Also
Reference
BlowfishCipher Class
Decipher Overload
CryptSharp.Utility Namespace
```

Copyright 2010-2014 James F. Bellinger

# BlowfishCipherDecipher Method (Byte, Int32, Byte, Int32)

CryptSharp

Reverses the encipherment of eight bytes of data from one buffer and places the result in another buffer.

Namespace: CryptSharp.Utility

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

```
C# VB C++ F#
                                                                 Copy
 public void Decipher(
            byte[] inputBuffer,
            int inputOffset,
            byte[] outputBuffer,
            int outputOffset
 )
Parameters
inputBuffer
  Type: SystemByte
  The buffer to read enciphered data from.
inputOffset
  Type: SystemInt32
  The offset of the first enciphered byte.
outputBuffer
  Type: SystemByte
  The buffer to write plaintext data to.
outputOffset
  Type: SystemInt32
  The offset at which to place the first plaintext byte.
```

### See Also

### Reference

BlowfishCipher Class

**Decipher Overload** 

# CryptSharp.Utility Namespace

# **BlowfishCipherDispose Method**

CryptSharp

Clears all memory used by the cipher.

Namespace: CryptSharp.Utility

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## **Syntax**

```
C# VB C++ F#
                                                          Copy
 public void Dispose()
Implements
```

**IDisposableDispose** 

### **See Also**

Reference

BlowfishCipher Class

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

# **BlowfishCipherEncipher Method**

CryptSharp

## **Overload List**

	Name	Description
=	Encipher(Byte, Int32)	Enciphers eight bytes of data in-place.
= 6	Encipher(UInt32, UInt32)	Enciphers eight bytes of data.
= 4	Encipher(Byte, Int32, Byte, Int32)	Enciphers eight bytes of data from one buffer and places the result in another buffer.

### Top

## See Also

Reference

BlowfishCipher Class

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

# BlowfishCipherEncipher Method (Byte, **Int32)**

CryptSharp

Enciphers eight bytes of data in-place.

Namespace: CryptSharp.Utility

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

```
C# VB C++ F#
                                                                 Copy
 public void Encipher(
            byte[] buffer,
            int offset
 )
Parameters
buffer
  Type: SystemByte
  The buffer containing the data.
offset
  Type: SystemInt32
  The offset of the first byte to encipher.
```

### See Also

### Reference

BlowfishCipher Class

**Encipher Overload** 

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

# BlowfishCipherEncipher Method (UInt32, UInt32)

CryptSharp

Enciphers eight bytes of data.

Namespace: CryptSharp.Utility

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

```
C# VB C++ F#
                                                                Copy
 public void Encipher(
            ref uint x1,
            ref uint xr
 )
Parameters
хl
  Type: SystemUInt32
  The first four bytes.
xr
  Type: SystemUInt32
  The last four bytes.
See Also
Reference
BlowfishCipher Class
Encipher Overload
CryptSharp.Utility Namespace
```

Copyright 2010-2014 James F. Bellinger

# BlowfishCipherEncipher Method (Byte, Int32, Byte, Int32)

CryptSharp

Enciphers eight bytes of data from one buffer and places the result in another buffer.

Namespace: CryptSharp.Utility

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

```
C# VB C++ F#
                                                                 Copy
 public void Encipher(
            byte[] inputBuffer,
            int inputOffset,
            byte[] outputBuffer,
            int outputOffset
 )
Parameters
inputBuffer
  Type: SystemByte
  The buffer to read plaintext data from.
inputOffset
  Type: SystemInt32
  The offset of the first plaintext byte.
outputBuffer
  Type: SystemByte
  The buffer to write enciphered data to.
outputOffset
  Type: SystemInt32
  The offset at which to place the first enciphered byte.
```

### See Also

### Reference

BlowfishCipher Class

**Encipher Overload** 

# CryptSharp.Utility Namespace

# **BlowfishCipher Properties**

CryptSharp

The BlowfishCipher type exposes the following members.

# **Properties**

	Name	Description
s s	BCryptLength	The number of bytes returned by BCrypt.
<b>≅</b> s	BCryptMagic	The special string used encrypted in the BCrypt algorithm.
	IsKeyWeak	A Blowfish key is weak if one of its S-boxes has a duplicate entry. See http://www.schneier.com/paper-blowfish-oneyear.html for more information.

### Top

### **See Also**

Reference

BlowfishCipher Class

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

## **BlowfishCipherBCryptLength Property**

CryptSharp

The number of bytes returned by BCrypt.

Namespace: CryptSharp.Utility

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## **Syntax**

```
public static int BCryptLength { get; }

Property Value
Type: Int32

See Also

Reference
BlowfishCipher Class
CryptSharp.Utility Namespace
```

Copyright 2010-2014 James F. Bellinger

# **BlowfishCipherBCryptMagic Property**

CryptSharp

The special string used encrypted in the BCrypt algorithm.

Namespace: CryptSharp.Utility

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

```
Syntax
```

```
C# VB C++ F#

Copy

public static string BCryptMagic { get; }

Property Value
Type: String

See Also

Reference
BlowfishCipher Class
CryptSharp.Utility Namespace
```

Copyright 2010-2014 James F. Bellinger

## **BlowfishCipherIsKeyWeak Property**

CryptSharp

A Blowfish key is weak if one of its S-boxes has a duplicate entry. See http://www.schneier.com/paper-blowfish-oneyear.html for more information.

Namespace: CryptSharp.Utility

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

```
public bool IsKeyWeak { get; }

Property Value
Type: Boolean

See Also

Reference
BlowfishCipher Class
CryptSharp.Utility Namespace
```

# **DesCipher Class**

CryptSharp

Performs low-level encryption and decryption using the DES cipher.

### **Inheritance Hierarchy**

SystemObject CryptSharp.UtilityDesCipher

Namespace: CryptSharp.Utility

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

```
C# VB C++ F#

Copy

public class DesCipher : IDisposable
```

The **DesCipher** type exposes the following members.

### **Methods**

	Name	Description	
<b>=♦S</b>	Create	Creates a DES cipher using the provided key.	
<b>≡</b>	Crypt	Crypts eight bytes of data in-place.	
<b>≡</b>	Decipher	Reverses the encipherment of eight bytes of data from one buffer and places the result in another buffer.	
<b>≡</b>	Dispose	Clears all memory used by the cipher.	
<b>≡</b>	Encipher	Enciphers eight bytes of data from one buffer and places the result in another buffer.	

### Top

### **See Also**

Reference

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

# **DesCipher Methods**

CryptSharp

The DesCipher type exposes the following members.

## **Methods**

	Name	Description	
<b>=♦S</b>	Create	Creates a DES cipher using the provided key.	
<b>≡</b>	Crypt	Crypts eight bytes of data in-place.	
<b>≡</b>	Decipher	Reverses the encipherment of eight bytes of data from one buffer and places the result in another buffer.	
<b>≡</b>	Dispose	Clears all memory used by the cipher.	
<b>≡</b>	Encipher	Enciphers eight bytes of data from one buffer and places the result in another buffer.	

### Top

## **See Also**

### Reference

DesCipher Class

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

# **DesCipherCreate Method**

CryptSharp

Creates a DES cipher using the provided key.

Namespace: CryptSharp.Utility

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

### Parameters

key

Type: SystemByte

The DES key. This must be eight bytes.

### **Return Value**

Type: DesCipher A DES cipher.

### See Also

Reference

DesCipher Class

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

## **DesCipherCrypt Method**

CryptSharp

Crypts eight bytes of data in-place.

Namespace: CryptSharp.Utility

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

```
C# VB C++ F#
                                                                   Copy
 public void Crypt(
            byte[] buffer,
            int offset,
            int iterations,
            int salt
 )
Parameters
buffer
  Type: SystemByte
  The buffer to crypt. For traditional DES crypt, this is zero-initialized.
offset
  Type: SystemInt32
  The offset into the buffer.
iterations
  Type: SystemInt32
  The number of iterations to run.
salt
  Type: SystemInt32
  The salt, up to 24 bits.
```

### See Also

#### Reference

DesCipher Class

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

Reverses the encipherment of eight bytes of data from one buffer and places the result in another buffer.

Namespace: CryptSharp.Utility

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

outputBuffer

Type: SystemByte

The buffer to write plaintext data to.

outputOffset

Type: SystemInt32

The offset at which to place the first plaintext byte.

### See Also

#### Reference

DesCipher Class

CryptSharp.Utility Namespace

# **DesCipherDispose Method**

CryptSharp

Clears all memory used by the cipher.

Namespace: CryptSharp.Utility

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

```
C# VB C++ F#

public void Dispose()

Implements
IDisposableDispose

See Also
```

Reference

DesCipher Class

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

Enciphers eight bytes of data from one buffer and places the result in another buffer.

Namespace: CryptSharp.Utility

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

Type: SystemByte

The buffer to write enciphered data to.

### outputOffset

Type: SystemInt32

The offset at which to place the first enciphered byte.

### **See Also**

### Reference

DesCipher Class

CryptSharp.Utility Namespace

# **EksBlowfishKeyExpansionFlags Enumeration**

CryptSharp

Modifiers for Expensive Key Schedule (EKS) Blowfish key expansion behavior.

Namespace: CryptSharp.Utility

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**



Copy

[FlagsAttribute]

public enum EksBlowfishKeyExpansionFlags

### **Members**

Member name	Value	Description
None	0	Default behavior.
EmulateCryptBlowfishSignExtensionBug	1	The EksBlowfish code in CryptSharp was in specification at http://static.usenix.org/event/usenix99/prove Many other BCrypt implementations, howe originating with the crypt_blowfish C imple crypt_blowfish had a sign extension bug the characters previous to any 8-bit character to never had this bug. However, for those who compatibility for old password databases crelibraries, I have added *support* for the bug this flag.

### **See Also**

Reference

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

Pbkdf2 Class CryptSharp

Implements the PBKDF2 key derivation function.

### **Inheritance Hierarchy**

SystemObject SystemMarshalByRefObject System.IOStream

CryptSharp.UtilityPbkdf2

Namespace: CryptSharp.Utility

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## **Syntax**

```
C# VB C++ F#

Copy

public class Pbkdf2 : Stream
```

The **Pbkdf2** type exposes the following members.

### **Constructors**

	Name	Description
<b>≡</b>	Pbkdf2	Creates a new PBKDF2 stream.

Top

### **Methods**

	Name	Description
<b>≡</b>	Close	Closes the stream, clearing memory and disposing of the HMAC algorithm. (Overrides StreamClose.)
<b>=♦S</b>	ComputeDerivedKey	Computes a derived key.
<b>≡</b>	Read(Int32)	Reads from the derived key stream.
<b>≅</b>	Read(Byte, Int32, Int32)	When overridden in a derived class, reads a sequence of bytes from the current stream and advances the position within the stream by the number of bytes read.  (Overrides StreamRead(Byte, Int32, Int32).)
=0	Seek	When overridden in a derived class, sets the position within the

current stream.
(Overrides StreamSeek(Int64, SeekOrigin).)

### Top

# **Properties**

Name	Description
Length	The maximum number of bytes that can be derived is 2\32-1 times the HMAC size.  (Overrides StreamLength.)
Position	The position within the derived key stream. (Overrides StreamPosition.)

### Top

### **Examples**

```
Computing a Derived Key

using System.Security.Cryptography;
using CryptSharp.Utility;

// Compute a 128-byte derived key using HMAC-SHA256,
byte[] derivedKey = Pbkdf2.ComputeDerivedKey(new HMA
```

# See Also

### Reference

CryptSharp.Utility Namespace

### Pbkdf2 Constructor

CryptSharp

Creates a new PBKDF2 stream.

Namespace: CryptSharp.Utility

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

```
public Pbkdf2(
     KeyedHashAlgorithm hmacAlgorithm,
     byte[] salt,
     int iterations
)
```

### **Parameters**

hmacAlgorithm

Type: System.Security.CryptographyKeyedHashAlgorithm

The HMAC algorithm to use, for example HMACSHA256. Make sure to set Key.

salt

Type: SystemByte

The salt. A unique salt means a unique PBKDF2 stream, even if the original key is identical.

iterations

Type: SystemInt32

The number of iterations to apply.

### **See Also**

### Reference

Pbkdf2 Class

CryptSharp.Utility Namespace

### Copyright 2010-2014 James F. Bellinger

The Pbkdf2 type exposes the following members.

## **Methods**

	Name	Description
<b>≡</b>	Close	Closes the stream, clearing memory and disposing of the HMAC algorithm.  (Overrides StreamClose.)
<b>=♦S</b>	ComputeDerivedKey	Computes a derived key.
<b>≡</b>	Read(Int32)	Reads from the derived key stream.
= <b>(</b>	Read(Byte, Int32, Int32)	When overridden in a derived class, reads a sequence of bytes from the current stream and advances the position within the stream by the number of bytes read.  (Overrides StreamRead(Byte, Int32, Int32).)
=•	Seek	When overridden in a derived class, sets the position within the current stream.  (Overrides StreamSeek(Int64, SeekOrigin).)

Top

# **See Also**

Reference

Pbkdf2 Class

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

### Pbkdf2Close Method

CryptSharp

Closes the stream, clearing memory and disposing of the HMAC algorithm.

Namespace: CryptSharp.Utility

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

# **Syntax**

```
C# VB C++ F#

Copy

public override void Close()
```

### See Also

Reference

Pbkdf2 Class

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

### Pbkdf2ComputeDerivedKey Method

CryptSharp

Computes a derived key.

Namespace: CryptSharp.Utility

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## **Syntax**

hmacAlgorithm

Type: System. Security. Cryptography Keyed Hash Algorithm

The HMAC algorithm to use, for example HMACSHA256. Make sure to set Key.

salt

Type: SystemByte

The salt. A unique salt means a unique derived key, even if the original key is identical.

iterations

Type: SystemInt32

The number of iterations to apply.

derivedKeyLength

Type: SystemInt32

The desired length of the derived key.

### **Return Value**

Type: Byte

The derived key.

### See Also

Reference

## Pbkdf2 Class CryptSharp.Utility Namespace

# Pbkdf2Read Method

CryptSharp

# **Overload List**

	Name	Description
<b></b>	Read(Int32)	Reads from the derived key stream.
<b></b>	Read(Byte, Int32, Int32)	When overridden in a derived class, reads a sequence of bytes from the current stream and advances the position within the stream by the number of bytes read. (Overrides StreamRead(Byte, Int32, Int32).)

### Top

## **See Also**

Reference

Pbkdf2 Class

CryptSharp.Utility Namespace

## Pbkdf2Read Method (Int32)

CryptSharp

Reads from the derived key stream.

Namespace: CryptSharp.Utility

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

## **Syntax**

```
C# VB C++ F#

public byte[] Read(
    int count
)
```

### Parameters

count

Type: SystemInt32

The number of bytes to read.

### **Return Value**

Type: Byte

Bytes from the derived key stream.

### See Also

Reference

Pbkdf2 Class

Read Overload

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

### Pbkdf2Read Method (Byte, Int32, Int32) CryptSharp

When overridden in a derived class, reads a sequence of bytes from the current stream and advances the position within the stream by the number of bytes read.

Namespace: CryptSharp.Utility

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

```
C# VB C++ F#

public override int Read(
          byte[] buffer,
          int offset,
          int count
)
```

#### **Parameters**

buffer

Type: SystemByte

An array of bytes. When this method returns, the buffer contains the specified byte array with the values between *offset* and (*offset* + *count* - 1) replaced by the bytes read from the current source.

offset

Type: SystemInt32

The zero-based byte offset in *buffer* at which to begin storing the data read from the current stream.

count

Type: SystemInt32

The maximum number of bytes to be read from the current stream.

### Return Value

Type: Int32

The total number of bytes read into the buffer. This can be less than the number of bytes requested if that many bytes are not currently available, or zero (0) if the end of the stream has been reached.

## **Exceptions**

Exception	Condition
ArgumentException	The sum of <i>offset</i> and <i>count</i> is larger than the buffer length.
ArgumentNullException	buffer is null.
ArgumentOutOfRangeException	offset or count is negative.
IOException	An I/O error occurs.
NotSupportedException	The stream does not support reading.
ObjectDisposedException	Methods were called after the stream was closed.

# **See Also**

Reference

Pbkdf2 Class

Read Overload

CryptSharp.Utility Namespace

### Pbkdf2Seek Method

CryptSharp

When overridden in a derived class, sets the position within the current stream.

Namespace: CryptSharp.Utility

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

```
C# VB C++ F#

public override long Seek(
          long offset,
          SeekOrigin origin
)
```

### **Parameters**

offset

Type: SystemInt64

A byte offset relative to the *origin* parameter.

origin

Type: System.IOSeekOrigin

A value of type SeekOrigin indicating the reference point used to obtain the new position.

### **Return Value**

Type: Int64

The new position within the current stream.

## **Exceptions**

Exception	Condition
IOException	An I/O error occurs.
NotSupportedException	The stream does not support seeking, such as if the stream is constructed from a pipe or console output.
ObjectDisposedException	Methods were called after the stream was closed.

### See Also

#### Reference

# Pbkdf2 Class CryptSharp.Utility Namespace

# **Pbkdf2 Properties**

CryptSharp

The Pbkdf2 type exposes the following members.

# **Properties**

Name	Description
Length	The maximum number of bytes that can be derived is 2^32-1 times the HMAC size. (Overrides StreamLength.)
Position	The position within the derived key stream. (Overrides StreamPosition.)

### Top

### See Also

Reference

Pbkdf2 Class

CryptSharp.Utility Namespace

# **Pbkdf2Length Property**

CryptSharp

The maximum number of bytes that can be derived is  $2^32-1$  times the HMAC size.

Namespace: CryptSharp.Utility

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

```
C# VB C++ F#

Copy

public override long Length { get; }

Property Value
Type: Int64

See Also

Reference
Pbkdf2 Class
CryptSharp.Utility Namespace
```

# **Pbkdf2Position Property**

CryptSharp

The position within the derived key stream.

Namespace: CryptSharp.Utility

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

```
C# VB C++ F#

Copy

public override long Position { get; set; }

Property Value
Type: Int64

See Also

Reference
Pbkdf2 Class
CryptSharp.Utility Namespace
```

Copyright 2010-2014 James F. Bellinger

### Salsa20Core Class

CryptSharp

Implements the Salsa20 hash function.

### **Inheritance Hierarchy**

SystemObject CryptSharp.UtilitySalsa20Core

Namespace: CryptSharp.Utility

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

```
C# VB C++ F#

Copy

public static class Salsa20Core
```

The **Salsa20Core** type exposes the following members.

### **Methods**

	Name	Description
∉ <b>\$</b> S	Compute	Applies the Salsa20 hash function. It maps a 16 element input to an output of the same size.

Top

### **See Also**

Reference

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

# **Salsa20Core Methods**

CryptSharp

The Salsa20Core type exposes the following members.

# **Methods**

	Name	Description
ĕ∳S	Compute	Applies the Salsa20 hash function. It maps a 16 element input to an output of the same size.

### Top

### See Also

#### Reference

Salsa20Core Class

CryptSharp.Utility Namespace

Applies the Salsa20 hash function. It maps a 16 element input to an output of the same size.

Namespace: CryptSharp.Utility

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

```
C# VB C++ F#
                                                                Copy
 public static void Compute(
            int rounds,
            uint[] input,
            int inputOffset,
            uint[] output,
            int outputOffset
 )
Parameters
rounds
  Type: SystemInt32
  The number of rounds. SCrypt uses 8.
input
  Type: SystemUInt32
  The input buffer.
inputOffset
  Type: SystemInt32
  The offset into the input buffer.
output
  Type: SystemUInt32
  The output buffer.
outputOffset
  Type: SystemInt32
  The offset into the output buffer.
```

### See Also

#### Reference

Salsa20Core Class CryptSharp.Utility Namespace

# **SCrypt Class**

CryptSharp

Implements the SCrypt key derivation function.

### **Inheritance Hierarchy**

SystemObject CryptSharp.UtilitySCrypt

Namespace: CryptSharp.Utility

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

```
C# VB C++ F#

Copy

public static class SCrypt
```

The **SCrypt** type exposes the following members.

### **Methods**

	Name	Description
<b>=\$S</b>	ComputeDerivedKey	Computes a derived key.
∉∳S	GetEffectivePbkdf2Salt	The SCrypt algorithm creates a salt which it then uses as a one- iteration PBKDF2 key stream with SHA256 HMAC. This method lets you retrieve this intermediate salt.
≡ <b>\$</b> S	GetStream	Creates a derived key stream from which a derived key can be read.

Top

### **See Also**

Reference

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

The SCrypt type exposes the following members.

# **Methods**

	Name	Description
<b>=\$S</b>	ComputeDerivedKey	Computes a derived key.
ĕ∳S	GetEffectivePbkdf2Salt	The SCrypt algorithm creates a salt which it then uses as a one- iteration PBKDF2 key stream with SHA256 HMAC. This method lets you retrieve this intermediate salt.
≡ <b>≬S</b>	GetStream	Creates a derived key stream from which a derived key can be read.

Top

### **See Also**

Reference

**SCrypt Class** 

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

Computes a derived key.

Namespace: CryptSharp.Utility

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

```
public static byte[] ComputeDerivedKey(
          byte[] key,
          byte[] salt,
          int cost,
          int blockSize,
          int parallel,
          Nullable<int> maxThreads,
          int derivedKeyLength
)
```

#### **Parameters**

key

Type: SystemByte

The key to derive from.

salt

Type: SystemByte

The salt. A unique salt means a unique SCrypt stream, even if the original key is identical.

cost

Type: SystemInt32

The cost parameter, typically a fairly large number such as 262144. Memory usage and CPU time scale approximately linearly with this parameter.

blockSize

Type: SystemInt32

The mixing block size, typically 8. Memory usage and CPU time scale approximately linearly with this parameter.

parallel

Type: SystemInt32

The level of parallelism, typically 1. CPU time scales approximately linearly with this parameter.

#### maxThreads

Type: SystemNullableInt32

The maximum number of threads to spawn to derive the key. This is limited by the *parallel* value. null will use as many threads as possible.

#### derivedKeyLength

Type: SystemInt32

The desired length of the derived key.

#### **Return Value**

Type: Byte

The derived key.

### **See Also**

#### Reference

**SCrypt Class** 

CryptSharp.Utility Namespace

### Copyright 2010-2014 James F. Bellinger

The SCrypt algorithm creates a salt which it then uses as a one-iteration PBKDF2 key stream with SHA256 HMAC. This method lets you retrieve this intermediate salt.

Namespace: CryptSharp.Utility

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

```
C# VB C++ F#
                                                              Copy
 public static byte[] GetEffectivePbkdf2Salt(
           byte[] key,
           byte[] salt,
           int cost,
           int blockSize,
           int parallel,
           Nullable<int> maxThreads
 )
Parameters
key
  Type: SystemByte
  The key to derive from.
salt
  Type: SystemByte
  The salt. A unique salt means a unique SCrypt stream, even if the original
  key is identical.
cost
  Type: SystemInt32
  The cost parameter, typically a fairly large number such as 262144.
```

Memory usage and CPU time scale approximately linearly with this parameter.

blockSize

Type: SystemInt32

The mixing block size, typically 8. Memory usage and CPU time scale approximately linearly with this parameter.

### parallel

Type: SystemInt32

The level of parallelism, typically 1. CPU time scales approximately linearly with this parameter.

#### maxThreads

Type: SystemNullableInt32

The maximum number of threads to spawn to derive the key. This is limited by the *parallel* value. null will use as many threads as possible.

#### **Return Value**

Type: Byte

The effective salt.

### See Also

#### Reference

**SCrypt Class** 

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

Creates a derived key stream from which a derived key can be read.

Namespace: CryptSharp.Utility

Type: SystemInt32

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

```
C# VB C++ F#
                                                                 Copy
 public static Pbkdf2 GetStream(
            byte[] key,
            byte[] salt,
            int cost,
            int blockSize,
            int parallel,
            Nullable<int> maxThreads
 )
Parameters
key
  Type: SystemByte
  The key to derive from.
salt
  Type: SystemByte
  The salt. A unique salt means a unique scrypt stream, even if the original
  key is identical.
cost
  Type: SystemInt32
  The cost parameter, typically a fairly large number such as 262144.
  Memory usage and CPU time scale approximately linearly with this
  parameter.
blockSize
  Type: SystemInt32
  The mixing block size, typically 8. Memory usage and CPU time scale
  approximately linearly with this parameter.
parallel
```

The level of parallelism, typically 1. CPU time scales approximately linearly with this parameter.

### maxThreads

Type: SystemNullableInt32

The maximum number of threads to spawn to derive the key. This is limited by the *parallel* value. null will use as many threads as possible.

#### **Return Value**

Type: Pbkdf2

The derived key stream.

### **See Also**

#### Reference

**SCrypt Class** 

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

# **SecureComparison Class**

CryptSharp

Provides comparison methods resistant to timing attack.

### **Inheritance Hierarchy**

SystemObject CryptSharp.UtilitySecureComparison

Namespace: CryptSharp.Utility

**Assembly:** CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

```
C# VB C++ F#

Copy

public static class SecureComparison
```

The **SecureComparison** type exposes the following members.

### **Methods**

	Name	Description
<b>=\$S</b>	Equals	Compares two strings in a timing-insensitive manner.

Top

### **See Also**

Reference

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

# **SecureComparison Methods**

CryptSharp

The SecureComparison type exposes the following members.

### **Methods**

	Name	Description
≡ <b>\$</b> S	Equals	Compares two strings in a timing-insensitive manner.

Top

### **See Also**

Reference

SecureComparison Class

CryptSharp.Utility Namespace

Copyright 2010-2014 James F. Bellinger

### SecureComparisonEquals Method

CryptSharp

Compares two strings in a timing-insensitive manner.

Namespace: CryptSharp.Utility

Assembly: CryptSharp (in CryptSharp.dll) Version: 2.1.0.0

### **Syntax**

```
public static bool Equals(
          string potentialAttackerSuppliedString,
          string referenceString
)
```

#### **Parameters**

potentialAttackerSuppliedString

Type: SystemString

The string controlled by a potential attacker.

referenceString

Type: SystemString

The string not controlled by a potential attacker.

#### **Return Value**

Type: Boolean

true if the strings are equal.

### **Remarks**

If the reference string is zero-length, this method does not protect it against timing attacks. If the reference string is extremely long, memory caching effects may reveal that fact.

### **See Also**

#### Reference

SecureComparison Class

CryptSharp.Utility Namespace

### Copyright 2010-2014 James F. Bellinger