**PowerBuilder加解密功能调用设计文档**

艾普阳科技(深圳)有限公司

2018

# 概述

## 目的

## 范围

## 定义

# 用户行为设计

## 编解码

* **Hex编解码**

Blob lblb\_data

String ls\_HexStr

lblb\_data = Blob("Test Hex", EncodingANSI!)

// 定义编解码器

CoderObject lnv\_CoderObject

lnv\_CoderObject = Create CoderObject

// Hex编码

ls\_HexStr = lnv\_CoderObject.HexEncode(lblb\_data)

// Hex解码

lblb\_data = lnv\_CoderObject.HexDecode(ls\_HexStr)

* **Base64编解码**

Blob lblb\_data

String ls\_Base64Str

lblb\_data = Blob("Test Base64", EncodingANSI!)

// 定义编解码器

CoderObject lnv\_CoderObject

lnv\_CoderObject = Create CoderObject

// Base64编码

ls\_Base64Str = lnv\_CoderObject.Base64Encode(lblb\_data)

// Base64解码

lblb\_data = lnv\_CoderObject.Base64Decode(ls\_Base64Str)

* **Url编解码**

Blob lblb\_data

String ls\_UrlStr

lblb\_data = Blob("https://github.com/AppeonTest/艾普阳", EncodingANSI!)

// 定义编解码器

CoderObject lnv\_CoderObject

lnv\_CoderObject = Create CoderObject

// Url编码

ls\_UrlStr = lnv\_CoderObject.UrlEncode(lblb\_data)

// Url解码

lblb\_data = lnv\_CoderObject.UrlDecode(ls\_UrlStr)

## 加解密

* **AES加解密**

Blob lblb\_data

Blob lblb\_key

Blob lblb\_iv

lblb\_data = Blob("Test AES", EncodingANSI!)

lblb\_key = Blob("Test Key12345678", EncodingANSI!)

lblb\_iv = Blob("Test IV 12345678", EncodingANSI!)

// 定义加解密器

CrypterObject lnv\_CrypterObject

lnv\_CrypterObject = Create CrypterObject

// AES加密

Blob lblb\_encrypt

lblb\_encrypt = lnv\_CrypterObject.SymmetricEncrypt(AES!, lblb\_data, lblb\_key, &

OperationModeCBC!, lblb\_iv, PKCSPadding!)

// AES 解密

Blob lblb\_decrypt

lblb\_decrypt = lnv\_CrypterObject.SymmetricDecrypt(AES!, lblb\_encrypt, lblb\_key, &

OperationModeCBC!, lblb\_iv, PKCSPadding!)

* **RSA加解密**

Blob lblb\_data

Blob lblb\_privKey

Blob lblb\_pubKey

lblb\_data = Blob("Test Rsa", EncodingANSI!)

// 定义加解密器

CrypterObject lnv\_CrypterObject

lnv\_CrypterObject = Create CrypterObject

// 生成密钥

lnv\_CrypterObject.AsymmetricGenerateKey(RSA!, 1024, lblb\_privKey, lblb\_pubKey)

// RSA加密

Blob lblb\_encrypt

lblb\_encrypt = lnv\_CrypterObject.AsymmetricEncrypt(RSA!, lblb\_data, lblb\_pubKey)

// RSA解密

Blob lblb\_decrypt

lblb\_decrypt = lnv\_CrypterObject.AsymmetricDecrypt(RSA!, lblb\_encrypt, lblb\_privKey)

* **RSA签名、验证**

Blob lblb\_data

Blob lblb\_privKey

Blob lblb\_pubKey

String ls\_HexStr

lblb\_data = Blob("Test Rsa", EncodingANSI!)

// 定义加解密器

CrypterObject lnv\_CrypterObject

lnv\_CrypterObject = Create CrypterObject

// 生成密钥

lnv\_CrypterObject.AsymmetricGenerateKey(RSA!, 1024, lblb\_privKey, lblb\_pubKey)

// RSA签名

Blob lblb\_signature

lblb\_signature = lnv\_CrypterObject.AsymmetricSign(RSA!, lblb\_data, lblb\_privKey)

// RSA 验证

Integer li\_isPass

li\_isPass = lnv\_CrypterObject.AsymmetricVerifySign(RSA!, lblb\_data, lblb\_pubKey, lblb\_signature)

* **MD5加密**

Blob lblb\_data

Blob lblb\_md5

lblb\_data = Blob("Test Hex", EncodingANSI!)

// 定义加解密器

CrypterObject lnv\_CrypterObject

lnv\_CrypterObject = Create CrypterObject

// MD5加密

lblb\_md5 = lnv\_CrypterObject.MD5(lblb\_data)

* **SHA加密**

Blob lblb\_data

Blob lblb\_sha1

lblb\_data = Blob("Test SHA1", EncodingANSI!)

// 定义加解密器

CrypterObject lnv\_CrypterObject

lnv\_CrypterObject = Create CrypterObject

// SHA加密

lblb\_sha1= lnv\_CrypterObject.SHA(SHA1!, lblb\_data)

* **HMAC加密**

Blob lblb\_data

Blob lblb\_key

Blob lblb\_hmac

lblb\_data = Blob("Test HMAC", EncodingANSI!)

lblb\_key = Blob("Test HMAC Key", EncodingANSI!)

// 定义加解密器

CrypterObject lnv\_CrypterObject

lnv\_CrypterObject = Create CrypterObject

// HMAC加密

lblb\_hmac= lnv\_CrypterObject.HMAC(HMACMD5!, lblb\_data, lblb\_key)

# 对象设计

## Enumeration(枚举)

### SymmetricAlgorithm

* **AES!**：The Advanced Encryption Standard.
* **DES!**：The Data Encryption Standard.
* **TDES!**：The Triple-DES.
* **DESX!**：The DES-XEX3.
* **Blowfish!**：The Blowfish.

### OperationMode

* **OperationModeECB!**：The Electronic Codebook(ECB) mode.
* **OperationModeCBC!**：The Cipher Block Chaining(CBC) mode.
* **OperationModeCFB!**：The Cipher Feedback(CFB) mode.
* **OperationModeOFB!**：The Output Feedback(OFB) mode.
* **OperationModeCTR!**：The Counter(CTR) mode.

### PaddingScheme

* **NoPadding!**：No padding added to a block.
* **ZerosPadding!**：0’s padding added to a block.
* **PKCSPadding!**：PKCS #5 padding added to a block.
* **OneAndZerosPadding!**：1 and 0’s padding added to a block.
* **DefaultPadding!**：Default padding scheme.DefaultPadding! means PKCSPadding! for ECB or CBC mode.Otherwise,NoPadding! for modes like CFB,OFB,CTR.

### AsymmetricAlgorithm

* **RSA!**：The Rivest-Shamir-Adleman cryptopsystem.
* **DSA!**：The Digital Signature Algorithm.
* **Rabin!：**The Rabin Algorithm.

### SHAAlgorithm

* **SHA1!**：SHA1.
* **SHA224!**：SHA224.
* **SHA256!**：SHA256.
* **SHA384!**：SHA384.
* **SHA512!**：SHA512.
* **SHA3\_224!**：SHA3-224.
* **SHA3\_256!**：SHA3-256.
* **SHA3\_384!**：SHA3-384.
* **SHA3\_512!**：SHA3-512.

### HMACAlgorithm

* **HMACMD5!**：HMAC-MD5.
* **HMACSHA1!**：HMAC-SHA1.
* **HMACSHA224!**：HMAC-SHA224.
* **HMACSHA256!**：HMAC-SHA256.
* **HMACSHA384!**：HMAC-SHA384.
* **HMACSHA512!**：HMAC-SHA512.

## CoderObject(编解码)

### HexEncode

**Description**

Encoding a blob with Hex encoder.

**Syntax**

String HexEncode(Blob variable)

**Argument**

* variable. A blob whose value is the data you want to encoding with Hex encoder.

**Return Value**

String. Returns the result of the encoding if it succeeds. If any argument’s value is null, the method returns null. If an error occurs, throw the exception.

### HexDecode

**Description**

Decoding a string with Hex decoder.

**Syntax**

Blob HexDecode(String variable)

**Argument**

* variable. A string whose value is the data you want to decoding with Hex decoder.

**Return Value**

Blob. Returns the result of the decoding if it succeeds. If any argument’s value is null, the method returns null. If an error occurs, throw the exception.

### Base64Encode

**Description**

Encoding a blob with Base64 encoder.

**Syntax**

String Base64Encode(Blob variable)

**Argument**

* variable. A blob whose value is the data you want to encoding with Base64 encoder.

**Return Value**

String. Returns the result of the encoding if it succeeds. If any argument’s value is null, the method returns null. If an error occurs, throw the exception.

### Base64Decode

**Description**

Decoding a string with Base64 decoder.

**Syntax**

Blob Base64Decode(String variable)

**Argument**

* variable. A string whose value is the data you want to decoding with Base64 decoder.

**Return Value**

Blob. Returns the result of the decoding if it succeeds. If any argument’s value is null, the method returns null. If an error occurs, throw the exception.

### UrlEncode

**Description**

Encoding a blob with Url encoder.

**Syntax**

String UrlEncode(Blob variable)

**Argument**

* variable. A blob whose value is the data you want to encoding with Url encoder.

**Return Value**

String. Returns the result of the encoding if it succeeds. If any argument’s value is null, the method returns null. If an error occurs, throw the exception.

### UrlDecode

**Description**

Decoding a string with Url decoder.

**Syntax**

Blob UrlDecode(String variable)

**Argument**

* variable. A string whose value is the data you want to decoding with Url decoder.

**Return Value**

Blob. Returns the result of the decoding if it succeeds. If any argument’s value is null, the method returns null. If an error occurs, throw the exception.

## CrypterObject(加解密)

### SymmetricEncrypt

**Description**

Encrypts a blob with symmetric algorithm.

**Syntax**

Blob SymmetricEncrypt(SymmetricAlgorithm algorithm, Blob variable, Blob key{, OperationMode operationmode{, Blob iv{, PaddingScheme padding}}})

**Argument**

* algorithm. A value of the symmetric algorithm enumerated type that specifies the type of symmetric algorithm. Values are:
  + AES! – The Advanced Encryption Standard
  + DES! – The Data Encryption Standard
  + TDES! – The Triple-DES
  + DESX! – The DES-XEX3
  + Blowfish! – The Blowfish
* variable. A blob whose value is the data you want to encrypt with symmetric algorithm.
* key. Blob specifying the secret key.

The length of the secret key can be 128 bits、192 bits、256 bits with AES.

The length of the secret key must be 64 bits with DES.

The length of the secret key can be 128 bits、192 bits with TDES.

The length of the secret key must be 192 bits with DESX.

The length of the secret key can be 32 bits~448 bits with Blowfish.

* operationmode(optional). A value of the OperationMode enumerated type that specifies the mode of operation. Values are:
  + OperationModeECB! – (Default)The Electronic Codebook(ECB) mode
  + OperationModeCBC! – The Cipher Block Chaining(CBC) mode
  + OperationModeCFB! – The Cipher Feedback(CFB) mode
  + OperationModeOFB! – The Output Feedback(OFB) mode
  + OperationModeCTR! – The Counter(CTR) mode
* iv(optional). Blob specifying the initialization vector. Zeros filled by default. In the AES algorithm, the effective length of the iv is 16 bytes and the others are 8 bytes. If the length is not enough, it will be automatically filled with zeros. In the ECB operation mode, the iv will be ignored.
* padding(optional). A value of the PaddingScheme enumerated type that specifies the padding schemes used for block cipher.Values are:
  + NoPadding! – No padding added to a block
  + ZerosPadding! – 0’s padding added to a block
  + PKCSPadding! – PKCS #5 padding added to a block
  + OneAndZerosPadding! – 1 and 0’s padding added to a block
  + DefaultPadding! – (Default)Default padding scheme.DefaultPadding! means PKCSPadding! for ECB or CBC mode.Otherwise,NoPadding! for modes like CFB,OFB,CTR.

ZerosPadding!、PKCSPadding!、OneAndZerosPadding! can be used with ECB and CBC operation mode.

NoPadding!、ZerosPadding! can be used with CFB、OFB and CTR operation mode.

**Return Value**

Blob. Returns the result of the encrypt if it succeeds. If any argument’s value is null, the method returns null. If an error occurs, throw the exception.

### SymmetricDecrypt

**Description**

Decrypts a blob with symmetric algorithm.

**Syntax**

Blob SymmetricDecrypt(SymmetricAlgorithm algorithm, Blob variable, Blob key{, OperationMode operationmode{, Blob iv{, PaddingScheme padding}}})

**Argument**

* algorithm. A value of the symmetric algorithm enumerated type that specifies the type of symmetric algorithm. Values are:
  + AES! – The Advanced Encryption Standard
  + DES! – The Data Encryption Standard
  + TDES! – The Triple-DES
  + DESX! – The DES-XEX3
  + Blowfish! – The Blowfish
* variable. A blob whose value is the data you want to decrypt with symmetric algorithm.
* key. Blob specifying the secret key.

The length of the secret key can be 128 bits、192 bits、256 bits with AES.

The length of the secret key must be 64 bits with DES.

The length of the secret key can be 128 bits、192 bits with TDES.

The length of the secret key must be 192 bits with DESX.

The length of the secret key can be 32 bits~448 bits with Blowfish.

* operationmode(optional). A value of the OperationMode enumerated type that specifies the mode of operation.Values are:
  + OperationModeECB! – (Default)The Electronic Codebook(ECB) mode
  + OperationModeCBC! – The Cipher Block Chaining(CBC) mode
  + OperationModeCFB! – The Cipher Feedback(CFB) mode
  + OperationModeOFB! – The Output Feedback(OFB) mode
  + OperationModeCTR! – The Counter(CTR) mode
* iv(optional). Blob specifying the initialization vector. Zeros filled by default. In the AES algorithm, the effective length of the iv is 16 bytes and the others are 8 bytes. If the length is not enough, it will be automatically filled with zeros. In the ECB operation mode, the iv will be ignored.
* padding(optional). A value of the PaddingScheme enumerated type that specifies the padding schemes used for block cipher.Values are:
  + NoPadding! – No padding added to a block
  + ZerosPadding! – 0’s padding added to a block
  + PKCSPadding! – PKCS #5 padding added to a block
  + OneAndZerosPadding! – 1 and 0’s padding added to a block
  + DefaultPadding! – (Default)Default padding scheme.DefaultPadding! means PKCSPadding! for ECB or CBC mode.Otherwise,NoPadding! for modes like CFB,OFB,CTR.

ZerosPadding!、PKCSPadding!、OneAndZerosPadding! can be used with ECB and CBC operation mode.

NoPadding!、ZerosPadding! can be used with CFB、OFB and CTR operation mode.

**Return Value**

Blob. Returns the result of the decrypt if it succeeds. If any argument’s value is null, the method returns null. If an error occurs, throw the exception.

### AsymmetricEncrypt

**Description**

Encrypts a blob with asymmetric algorithm.

**Syntax**

Blob AsymmetricEncrypt(AsymmetricAlgorithm algorithm, Blob variable, Blob pubKey)

**Argument**

* algorithm. A value of the asymmetric algorithm enumerated type that specifies the type of asymmetric algorithm. Values are:
  + RSA! – The Rivest-Shamir-Adleman cryptopsystem
  + Rabin! – The Rabin Algorithm
* variable. A blob whose value is the data you want to encrypt with Public-Key cipher.
* pubKey. Blob specifying the public key.

**Return Value**

Blob. Returns the result of the encrypt if it succeeds. If any argument’s value is null, the method returns null. If an error occurs, throw the exception.

### AsymmetricDecrypt

**Description**

Decrypts a blob with asymmetric algorithm.

**Syntax**

Blob AsymmetricDecrypt(AsymmetricAlgorithm algorithm, Blob variable, Blob privKey)

**Argument**

* algorithm. A value of the asymmetric algorithm enumerated type that specifies the type of asymmetric algorithm. Values are:
  + RSA! – The Rivest-Shamir-Adleman cryptopsystem
  + Rabin! – The Rabin Algorithm
* variable. A blob whose value is the data you want to decrypt with Public-Key cipher.
* privKey. Blob specifying the private key.

**Return Value**

Blob. Returns the result of the decrypt if it succeeds. If any argument’s value is null, the method returns null. If an error occurs, throw the exception.

### AsymmetricSign

**Description**

Calculating the signature of the data with asymmetric algorithm.

**Syntax**

Blob AsymmetricSign(AsymmetricAlgorithm algorithm, Blob variable, Blob privKey)

**Argument**

* algorithm. A value of the asymmetric algorithm enumerated type that specifies the type of asymmetric algorithm. Values are:
  + RSA! – The Rivest-Shamir-Adleman cryptopsystem
  + DSA! – The Digital Signature Algorithm
  + Rabin! – The Rabin Algorithm
* variable. A blob whose value is the data you want to sign with Public-Key cipher.
* privKey. Blob specifying the private key.

**Return Value**

Blob. Returns the signature of the data if it succeeds. If any argument’s value is null, the method returns null. If an error occurs, throw the exception.

### AsymmetricVerifySign

**Description**

Verify the signature of the data with asymmetric algorithm.

**Syntax**

Integer AsymmetricVerifySign(AsymmetricAlgorithm algorithm, Blob variable, Blob pubKey, Blob sign)

**Argument**

* algorithm. A value of the asymmetric algorithm enumerated type that specifies the type of asymmetric algorithm. Values are:
  + RSA! – The Rivest-Shamir-Adleman cryptopsystem
  + DSA! – The Digital Signature Algorithm
  + Rabin! – The Rabin Algorithm
* variable. A blob whose value is the data you want to verify with Public-Key cipher.
* pubKey. Blob specifying the public key.
* sign. Blob specifying the signature.

**Return Value**

Integer. Returns 1 if it verify succeeds and -1 if it verify failed. If any argument’s value is null, the method returns null. If an error occurs, throw the exception.

### AsymmetricGenerateKey

**Description**

Generate secret key for asymmetric algorithm.

**Syntax**

Integer AsymmetricGenerateKey(AsymmetricAlgorithm algorithm, Integer len, ref Blob privKey, ref Blob pubKey)

**Argument**

* algorithm. A value of the asymmetric algorithm enumerated type that specifies the type of asymmetric algorithm. Values are:
  + RSA! – The Rivest-Shamir-Adleman cryptopsystem
  + DSA! – The Digital Signature Algorithm
  + Rabin! – The Rabin Algorithm
* len. Integer specifying the key length.
* privKey. Blob specifying the private key.
* pubKey. Blob specifying the public key.

**Return Value**

Integer. Returns 1 if it succeeds and -1 if it failed. If any argument’s value is null, the method returns null. If an error occurs, throw the exception.

### MD5

**Description**

Calculating the MD5 value of a blob.

**Syntax**

Blob MD5(Blob variable)

**Argument**

* variable. A blob whose value is the data you want to process with MD5.

**Return Value**

Blob. Returns the result of the MD5 if it succeeds. If any argument’s value is null, the method returns null. If an error occurs, throw the exception.

### SHA

**Description**

Calculating the SHA value of a blob.

**Syntax**

Blob SHA(SHAAlgorithm algorithm, Blob variable)

**Argument**

* algorithm. the hash algorithm type.
  + SHA1! – SHA1
  + SHA224! – SHA224
  + SHA256! – SHA256
  + SHA384! – SHA384
  + SHA512! – SHA512
  + SHA3\_224! – SHA3-224
  + SHA3\_256! – SHA3-256
  + SHA3\_384! – SHA3-384
  + SHA3\_512! – SHA3-512
* variable. A blob whose value is the data you want to process with SHA.

**Return Value**

Blob. Returns the result of the SHA if it succeeds. If any argument’s value is null, the method returns null. If an error occurs, throw the exception.

### HMAC

**Description**

Calculating the HMAC value of a blob.

**Syntax**

Blob HMAC(HMACAlgorithm algorithm, Blob variable, Blob key)

**Argument**

* algorithm. the HMAC algorithm type.
  + HMACMD5! – HMAC-MD5
  + HMACSHA1! – HMAC-SHA1
  + HMACSHA224! – HMAC-SHA224
  + HMACSHA256! – HMAC-SHA256
  + HMACSHA384! – HMAC-SHA384
  + HMACSHA512! – HMAC-SHA512
* variable. A blob whose value is the data you want to process with HMAC.
* key. Blob specifying the secret key.

**Return Value**

Blob. Returns the result of the HMAC if it succeeds. If any argument’s value is null, the method returns null. If an error occurs, throw the exception.

# 代码设计

此处设计包括两部分：

PBCrypt LIB库 ：为从Crypt DLL库适配到PowerBuilder System Function的适配器。

Crypt DLL库：为第三方加密库的适配器。

## 总体模块结构



## 子模块：PB对象

见“对象设计”。

## 子模块：PBCrypt LIB

### 模块交互

如图4-1所示，PBCryptLib调用Crypt.dll的ICrypt接口，实现PB系统函数。



图4-1 PBCrypt LIB组件

### 代码结构

其内部类的调用关系如图4-2所示：



图4-2 PBCrypt LIB库内部结构

PBCrypt LIB函数接口调用Crypt DLL导出接口类ICrypt中的对应方法，实现对应功能，从而实现PB系统函数功能。

### 接口定义

|  |  |  |
| --- | --- | --- |
| 类 | 接口 | 说明 |
| 标准 C 函数接口 | fnHexEncode | Hex编码 |
| fnHexDecode | Hex解码 |
| fnBase64Encode | Base64编码 |
| fnBase64Decode | Base64解码 |
| fnUrlEncode | Url编码 |
| fnUrlDecode | Url解码 |
| fnSymmetricEncrypt | 对称加密 |
| fnSymmetricDecrypt | 对称解密 |
| fnAsymmetricEncrypt | 非对称加密 |
| fnAsymmetricDecrypt | 非对称解密 |
| fnAsymmetricSign | 签名 |
| fnAsymmetricVerifySign | 验证 |
| fnAsymmetricGenerateKey | 生成一对密钥 |
| fnMD5 | MD5加密 |
| fnSHA | SHA加密 |
| fnHMAC | HMAC加密 |

## 子模块：Crypt DLL

### 模块交互

Crypt通用库DLL部分，封装第三方加密库，如图4-3所示，其对外导出一个接口类和创建及销毁该类实例的C函数。



图4-3 Crypt通用库

### 代码结构

Crypt库内部类之间的关系如图 4-4所示：



图4-4 Crypt通用库内部结构

Crypt通用库实现部分（DLL库）分三个层次实现，分别是：

**接口层**：对外导出接口：ICoderObject 、ICrypterObject。其中

* ICoderObject的主要职责：对数据进行编解码。
* ICrypterObject的主要职责：对数据进行单向加密、对称加解密、非对称加解密，及获取密钥。

**业务层**：由CCoderObjectAdapter、CCrypterObjectAdapter类实现。主要实现各种请求方法的调用逻辑。此类不直接调用底层加密相关的API接口。

**实现层**：此层通过调用开源的第三方加解密库（可以用Crypto++、libtomcrypt）来实现加解密功能。

### 接口定义

|  |  |  |
| --- | --- | --- |
| 类 | 接口 | 说明 |
| 标准 C 函数接口 | CreateCoderObject | 创建ICoderObject的实例 |
| CreateCrypterObject | 创建ICrypterObject的实例 |
| ICoderObject | HexEncode | Hex编码 |
| HexDecode | Hex解码 |
| Base64Encode | Base64编码 |
| Base64Decode | Base64解码 |
| UrlEncode | Url编码 |
| UrlDecode | Url解码 |
| Release | 销毁ICoderObject的实例 |
| ICrypterObject | SymmetricEncrypt | 对称加密 |
| SymmetricDecrypt | 对称解密 |
| AsymmetricEncrypt | 非对称加密 |
| AsymmetricDecrypt | 非对称解密 |
| SignMessage | 签名 |
| VerifyMessage | 验证 |
| GenerateKey | 生成一对密钥 |
| MD5 | MD5加密 |
| SHA | SHA加密 |
| HMAC | HMAC加密 |
| Release | 销毁ICrypterObject的实例 |