nirva

Nirva Crypt Service

Document Version: 1.05

nirva

Table of Contents

nirva

Overview

The CRYPT service is a NIRVA external service that provides commands for cryptography management using the algorithms AES, DES and RSA.

We use three modes STRING, URL and FILE. The FILE mode provides file encoding.
STRING and URL mode provide string encoding. In URL mode we escape some URL reserved characters.

Performance

The service is based upon a JAVA library.

nirva

Installation

The CRYPT service is delivered as a NIRVA package and can be installed like any NIRVA service directly from the NIRVA configuration web site. Please see the NIRVA configuration chapter in the NIRVA user's guide for further information.

Licensing

The CRYPT service is a licensed product.

The license can be installed like any NIRVA licenses. Please consult the NIRVA license chapter in the NIRVA user's guide for further information about license installation.

nirva

Configuration

There is no specific configuration.

Reference

This chapter gives the complete reference of all the CRYPT service commands.

Classes

Here are the available CRYPT service classes:

Class	Description
AES	AES algorithm commands
DES	DES algorithm commands
RSA	RSA algorithm commands
CHECKSUM	CHECKSUM algorithm commands

Error codes

The same error codes are shared with the four classes.

AES Class

Value	Description
101	ERROR_ILLEGAL_BLOCK
102	ERROR_BADPADDING
103	ERROR_INVALIDKEY
104	ERROR_INSTANCE
105	ERROR_CIPHER
106	ERROR_ALGORITHM

DES Class

Value	Description
101	ERROR_ILLEGAL_BLOCK
102	ERROR_BADPADDING
103	ERROR_INVALIDKEY
104	ERROR_INSTANCE
105	ERROR_CIPHER
106	ERROR_ALGORITHM

RSA Class

Value	Description
101	ERROR_ILLEGAL_BLOCK
102	ERROR_BADPADDING
103	ERROR_INVALIDKEY
104	ERROR_INSTANCE
105	ERROR_CIPHER
106	ERROR_ALGORITHM

CHECKSUM Class

Value	Description
101	ERROR_ILLEGAL_BLOCK
102	ERROR_BADPADDING
103	ERROR_INVALIDKEY
104	ERROR_INSTANCE
105	ERROR_CIPHER
106	ERROR_ALGORITHM

Commands

For each command, the reference gives the command name, the sources for which the command may be used, the command description, the eventual command permissions, the parameter list and the eventual list of objects created by the command.

The parameters described in this chapter are command specific parameters. For general parameters, please refer to the Nirva command syntax chapter.

The available sources are:

- Client for all Nirva client interfaces including Nirva client library (nvc).
- Web for commands from a web browser.
- Procedure for commands from a Nirva procedure.
- Service for commands from service to service

AES class

This class is used to encode and decode using AES algorithm.

ENCODE

CRYPT:AES:ENCODE

Source	Use Input Container	Use Output Container
Client Web Procedure Service	YES	YES

Description

This command encodes a given content using a specified key. It creates an object in the output container which name is specified by the RESULT parameter.

Parameters

MODE	STRING, FILE or URL
CONTENT	Name of the NIRVA object in FILE mode, string content otherwise
KEY	String content
CIPHER_MODE	Default is ECB
CIPHER_PADDING	Default is PKCS5Padding
VECTOR	Initialisation vector of 16 bytes string.
RESULT	Name of the result object (same type as mode)
BASE	Base64 or 16 (default)

Permissions

None

Objects created

RESULT	This is a Nirva FILE(or STRING) object containing the result of encoding

---

DECODE

CRYPT:AES:DECODE

Source	Use Input Container	Use Output Container
Client Web Procedure Service	YES	YES

Description

This command decodes a given content using a specified key. It creates an object in the output container which name is specified by the RESULT parameter.

Parameters

MODE	STRING, FILE or URL
CONTENT	Name of the NIRVA object in FILE mode, string content otherwise
KEY	String content

CIPHER_MODE	Default is ECB
CIPHER_PADDING	Default is PKCS5Padding
VECTOR	Initialisation vector of 16 bytes string.
RESULT	Name of the result object (same type as mode)
BASE	Base64 or 16 (default)

Permissions

None

Objects created

RESULT	This is a Nirva FILE(or STRING) object containing the result of encoding

DES class

This class is used to encode and decode using DES algorithm.

ENCODE

CRYPT:DES:ENCODE

Source	Use Input Container	Use Output Container
Client Web Procedure Service	YES	YES

Description

This command encodes a given content using a specified key. It creates an object in the output container which name is specified by the RESULT parameter.

Parameters

MODE	STRING, FILE or URL
CONTENT	Name of the NIRVA object in FILE mode, string content otherwise
KEY	String content

CIPHER_MODE	Default is ECB
CIPHER_PADDING	Default is PKCS5Padding
VECTOR	Initialisation vector of 16 bytes string.
RESULT	Name of the result object (same type as mode)
BASE	Base64 or 16 (default)

Permissions

None

Objects created

RESULT	This is a Nirva FILE(or STRING) object containing the result of encoding

---

DECODE

CRYPT:DES:DECODE

Source	Use Input Container	Use Output Container
Client Web Procedure Service	YES	YES

Description

This command decodes a given content using a specified key. It creates an object in the output container which name is specified by the RESULT parameter.

Parameters

MODE	STRING, FILE or URL
CONTENT	Name of the NIRVA object in FILE mode, string content otherwise
KEY	String content
CIPHER_MODE	Default is ECB
CIPHER_PADDING	Default is PKCS5Padding
VECTOR	Initialisation vector of 16 bytes string.
RESULT	Name of the result object (same type as mode)

BASE	Base64 or 16 (default)

Permissions

None

Objects created

RESULT	This is a Nirva FILE(or STRING) object containing the result of encoding

RSA class

This class is used to encode and decode using RSA algorithm.

GENERATE_KEYS

CRYPT:RSA:GENERATE_KEYS

Source	Use Input Container	Use Output Container
Client Web Procedure Service	YES	YES

Description

This command creates a pair of private and public keys. The result is stored in three string objects PUBLIC_KEY_EXP, KEY_MOD and PRIVATE_KEY_EXP

Parameters

Permissions

None

Objects created

PUBLIC_KEY_EXP	This is a Nirva STRING object containing the exponent of the public key
PRIVATE_KEY_EXP	This is a Nirva STRING object containing the exponent of the private key
KEY_MOD	This is a Nirva STRING object containing the modulus of the keys

ENCODE

CRYPT:RSA:ENCODE

Source	Use Input Container	Use Output Container
Client Web Procedure Service	YES	YES

Description

This command encodes a given content using a specified key. It creates an object in the output container which name is specified by the RESULT parameter.

Parameters

MODE	STRING, FILE or URL
CONTENT	Name of the NIRVA object in FILE mode, string content otherwise
KEY_MOD	String content
KEY_EXP	String content
RESULT	Name of the result object (same type as mode)
BASE	Base64 or 16 (default)

Permissions

None

Objects created

RESULT	This is a Nirva FILE(or STRING) object containing the result of encoding

DECODE

CRYPT:RSA:DECODE

Source	Use Input Container	Use Output Container

Source	Use Input Container	Use Output Container
Client Web Procedure Service	YES	YES

Description

This command decodes a given content using a specified key. It creates an object in the output container which name is specified by the RESULT parameter.

Parameters

MODE	STRING, FILE or URL
CONTENT	Name of the NIRVA object in FILE mode, string content otherwise
KEY_MOD	String content
KEY_EXP	String content
RESULT	Name of the result object (same type as mode)
BASE	Base64 or 16 (default)

Permissions

None

Objects created

RESULT	This is a Nirva FILE(or STRING) object containing the result of encoding

CHECKSUM class

This class is used to encode and decode using CHEKCSUM methods.

MD5

CRYPT:CHECKSUM:MD5

Source	Use Input Container	Use Output Container

Source	Use Input Container	Use Output Container
Client Web Procedure Service	YES	YES

Description

This command creates a checksum of a NIRVA File or String.

Parameters

MODE	STRING, FILE or URL
CONTENT	Name of the NIRVA object in FILE mode, string content otherwise
RESULT	Name of the result object (same type as mode)
BASE	Base64 or 16 (default)

Permissions

None

Objects created

RESULT	This is a Nirva FILE(or STRING) object containing the result of encoding

---

SHA1

CRYPT:CHECKSUM:SHA1

Source	Use Input Container	Use Output Container
Client Web Procedure Service	YES	YES

Description

This command creates a checksum of a NIRVA File or String.

Parameters

MODE	STRING, FILE or URL
CONTENT	Name of the NIRVA object in FILE mode, string content otherwise
RESULT	Name of the result object (same type as mode)
BASE	Base64 or 16 (default)

Permissions

None

Objects created

RESULT	This is a Nirva FILE(or STRING) object containing the result of encoding

nirva

URL escaped caracters

Table list

Name	Character	Encoding
Dollar	$	%24
Ampersand	&	%26
Plus	+	%2B
Comma	,	%2C
Slash	/	%2F
Colon	:	%3A
Semi-colon	;	%3B
Equals	=	%3D
QuestionM	?	%3F
At	@	%40
Space		%20
QuotationM	‘	%27
Less than	<	%3C
Greater than	>	%3E
Pound	#	%23
Percent	%	%25
Left brace	{	%7B
Right brace	}	%7D
Back slash	\	%5C
Caret	^	%5E
Tilde	~	%7B
Left	[	%5B
Right	]	%5D
Grave	`	%60

nirva

Java Cryptography architecture

Overview

This is a simple extract from the java web site.

The following names can be specified as the *algorithm* component in a transformation when requesting an instance of **Cipher**:

• **AES**: Advanced Encryption Standard as specified by NIST in a draft FIPS. Based on the Rijndael algorithm by Joan Daemen and

Vincent Rijmen, AES is a 128-bit block cipher supporting keys of 128, 192, and 256 bits.

• **ARCFOUR/RC4**: A stream cipher developed by Ron Rivest. For more information, see K. Kaukonen and R. Thayer, "A Stream Cipher

Encryption Algorithm 'Arcfour'", Internet Draft (expired), draft-kaukonen-cipher-arcfour-03.txt.

• **Blowfish**: The block cipher designed by Bruce Schneier.

• **DES**: The Digital Encryption Standard as described in FIPS PUB 46-2.

• **DESede**: Triple DES Encryption (DES-EDE).

• **ECIES (Elliptic Curve Integrated Encryption Scheme)**

• **PBEWith<digest>And<encryption>** or **PBEWith<prf>And<encryption>**: The password-based encryption algorithm (PKCS #5), using

the specified message digest (<digest>) or pseudo-random function (<prf>) and encryption algorithm (<encryption>). Examples:

• **PBEWithMD5AndDES**: The password-based encryption algorithm as defined in: RSA Laboratories, "PKCS #5: Password-

Based Encryption Standard," version 1.5, Nov 1993. Note that this algorithm implies *CBC* as the cipher mode and

PKCS5Padding as the padding scheme and cannot be used with any other cipher modes or padding schemes.

• **PBEWithHmacSHA1AndDESede**: The password-based encryption algorithm as defined in: RSA Laboratories, "PKCS #5:

Password-Based Cryptography Standard," version 2.0, March 1999.

• **RC2**, **RC4**, and **RC5**: Variable-key-size encryption algorithms developed by Ron Rivest for RSA Data Security, Inc.

• **RSA**: The RSA encryption algorithm as defined in PKCS #1.

Mode

The following names can be specified as the *mode* component in a transformation when requesting an instance of **Cipher**:

• **NONE**: No mode.

• **CBC**: Cipher Block Chaining Mode, as defined in FIPS PUB 81.

• **CFB**: Cipher Feedback Mode, as defined in FIPS PUB 81.

• **ECB**: Electronic Codebook Mode, as defined in: The National Institute of Standards and Technology (NIST) Federal Information

Processing Standard (FIPS) PUB 81, "DES Modes of Operation," U.S. Department of Commerce, Dec 1980.

• **OFB**: Output Feedback Mode, as defined in FIPS PUB 81.

• **PCBC**: Propagating Cipher Block Chaining, as defined by Kerberos V4.

Padding

The following names can be specified as the *padding* component in a transformation when requesting an instance of **Cipher**:

• **ISO10126Padding**. This padding for block ciphers is described in 5.2 Block Encryption Algorithms in the W3C's "XML Encryption Syntax

and Processing" document.

• **NoPadding**: No padding.

• **OAEPWith<digest>And<mgf>Padding**: Optimal Asymmetric Encryption Padding scheme defined in PKCS #1, where <digest> should

be replaced by the message digest and <mgf> by the mask generation function. Example: OAEPWithMD5AndMGF1Padding.

• **PKCS5Padding**: The padding scheme described in: RSA Laboratories, "PKCS #5: Password-Based Encryption Standard," version 1.5,

November 1993.

• **SSL3Padding**: The padding scheme defined in the SSL Protocol Version 3.0, November 18, 1996, section 5.2.3.2 (CBC block cipher):

block-ciphered struct {

opaque content[SSLCompressed.length];

opaque MAC[CipherSpec.hash_size];

uint8 padding[GenericBlockCipher.padding_length];

uint8 padding_length;

} GenericBlockCipher;

The size of an instance of a GenericBlockCipher must be a multiple of the block cipher's block length.

The padding length, which is always present, contributes to the padding, which implies that if:

sizeof(content) + sizeof(MAC) % block_length = 0,

padding has to be (block_length - 1) bytes long, because of the existence of padding_length.

This make the padding scheme similar (but not quite) to PKCS5Padding, where the padding length is encoded in the padding (and

ranges from 1 to block_length). With the SSL scheme, the sizeof(padding) is encoded in the always present padding_length and

therefore ranges from 0 to block_length-1.

Note that this padding mechanism is not supported by the "SunJCE" provider.