I'm not robot

reCAPTCHA

Continue

Encoding base64 pdf

Encoding base64 decode. Encoding base64 apex. Encoding base64 in java. Encoding base64 linux. Encoding base64 python. Encoding base64 javascript. Encoding base64 c#. Encoding base64 golang.

Support for encoding and decoding base64. This module provides two default implementations of Base64 Encoding, Base64 alphabet with a standard coding and BASE64URL variant that has a modified coding alphabet designed to be safe for embedding in URLs and file names. Both variants are implemented as instantiations of BASE64Impl model. The majority of users you must not use this template directly; However, it can be used to create custom coding base64, like the one that omits the padding characters or one that is sure to be incorporated within a regular expression. Example UBYTE [] data = [0x14, 0xfb, 0x9C, 0x03, 0xd9, 0x7e]; CONST (CHAR) [] = base64.code encoded (data); Assert (encodificato == "fpuuca9l +"); ubyte [] = Base64.decode decoded ("fpuuca9l +"); Assert (decoded == [0x14, 0xfb, 0x9C, 0x03, 0xd9, 0x7e]); The API of the range is supported for both encoding and decoding: Sample Files F = file ("./ text.txt", "r"); Scope (output) f.close (); Appender! String = MIME64 Antepender! String; foreach (encoded; base64.encoder (f.ychunk (57))) {mime64.put (encoded); mime64.put ("\r"); } Writeln (mime64.data); References RFC 4648 - The licensing Base16, Base32, and Base64 Encoding data: = alias base64 base64Impl! ('+', '/', '='); Encoding Implementation standard64 base64. See base64Impl for a description of the available methods. Examples: UBYTE [] data = [0x83, 0xD7, 0x30, 0x7a, 0x01, 0x3f]; writeln (base64.code (data)); writeln (Base64.decode ("g9cwege /")); aka base64url = base64Impl! ('-', '_', '='); Change in the base64 encoding that is safe for use in URLs and file names. See base64Impl for a description of the available methods. Examples: UBYTE [] data = [0x83, 0xD7, 0x30, 0x7a, 0x01, 0x3f]; writeln (base64url.code (data)); writeln (base64url.decode ("g9cwege ")); aka base64urlnopadding = base64Impl! ('-', '_', '\X00'); Change not used the base64 encoding that is safe for use in URLs and file names, as used in CSFR 4648 and 7515 (JWS / JWT / JWE). See base64Impl for a description of the available methods. Examples: UBYTE [] data = [0x83, 0xD7, 0x30, 0x7b, 0xEF]; writeln (base64urlnopadding.code (data)); writeln (base64urlnopadding.decode "G9Cwe-8")); Template Base64Impl (Map62th Char, Char Map63th, Charding = '=') Model for the implementation of base64 encoding and decoding. For most purposes, the direct use of this model is not necessary; Instead, this module provides default implementations: Base64, implementing the encoding of base64 Base64 and Base64url and Base64urlnopadding, that implement the base64 variant to be used in URLs and file names, with and without padding, respectively. The custom BASE64 encoding schemes can be implemented by instantiating this model with appropriate arguments. For example: alias = base64re base64Impl! ('!', '=', Base64.Nopadding); Note encoded strings will have no padding if the padding parameter is set to Nopadding. Examples: import std.string: representation; ubyte [] squarerr; writeln (base64.code (empty)); writeln (base64.code ("f" .representation)); writeln (base64.code ("foo" .representation); aka base64re = base64Impl! ('!', '=', Base64.Nopadding); writeln (base64re.code ("f" .representation)); writeln (base64re.code ("foo" .representation)); It represents encoding without padding PURE nothrow @SAFE TAGLIO_T ENCODELLAINGHINGHINGHING (IN TIZE_T AUTOLENGENGUEZIONE); Compute the length needed to store the encoded string corresponding to an input of given length. Parameters: taglia_t length of the length of sourceleng. The source array. Returns: the length of a base64 encoding of the length of a given matrix. Examples: ubyte [] data = [0x1a, 0x2b, 0x3c, 0x4D, 0x5D, Auto BUF = New Char [BASE64.CENCODELINGHINGHING (DATA.Longqual)]; writeln (base64.code (data, buf); Writhn (BUF); Pure @Rustued Char [] encoding (R1, R2) (in source R1, R2) SE (ISARRAY! R1 && Â¯ (elementsType! R1: Ubyte) && Â¯ (R2 == Char [])); Char [] encoding (R1, R2) (Source R1, buffer R2) If (! Israry! R1 && Isinputrange! R1: Ubyte) && Haslengenge! R1 && Â¯ (R2 == Char []))); Encoding the source in a char buffer [] using base64 base64 Parameters: R1 The source input field to encode. R2 char buffer [] buffer to store the encoded result. Returns: The buffer slice containing the encoded string. Examples: UBYTE [] data = [0x83, 0xD7, 0x30, 0x7a, 0x01, 0x3f]; char [32] buffer; Auto encodedLength = base64.encodeLength (data.length); assert (buffer.length> = encodedLength); Auto coded = Base64.encode (data, buffer []); assert (is encoded buffer [0 .. encodedLength]); writeln (encoded); encode size t (E, R) (const capacity (E) [] source ref R range car) if (it's (E: UBYTE !!) && isOutputRange (R, char) && is (R == char [])); !!! Encode size t (R1, R2) (source R1, R2 auto ref range) if (R1 IsArray && && isInputRange R1 is (ElementType R1: UBYTE !!!) && && hasLength R1 is (R2 == char []) && isOutputRange (R2, char)) !; Returns: The number of times put method of the output range has been invoked. Examples: import std.array: appender; car production appender = string (!); ubyte [] = data [0x1a, 0x2b, 0x3C, 0x4D, 0x5D, 0x6E]; writeln (Base64.encode (data output)); writeln (output.data); pure @safe char [] encode (Range) (Range source) if (IsArray isInputRange Gamma Gamma && && is (ElementType Range: UBYTE !!!) UBYTE) && hasLength Range); source coding to the newly allocated buffer. This convenient method reduces the need to manually handle the output buffer. Parameters: Source range The input range to code. Returns: A newly allocated char [] buffer containing the encoded string. Examples: ubyte [] data = [0x1a, 0x2b, 0x3c, 0x4D, 0x5D, 0x6E]; writeln (base64.code (data)); struct encoder (Range) if (isInputRange Range && is (ElementType Range: const (UBYTE) []) || is (ElementType Range: !!! const (char) []))); This range will be a forward if the underlying data source is at least one forward gear. Note This struct is not intended to be created in user code directly; instead use the encoder function. @property @trusted bool empty (); Returns: true if there are no more data encoded on the left. nothrow @property @safe char [] front (); Returns: The current encoded data block. Advancing the interval to the next encoded data block. Throws: Base64Exception If called when `empty` returns true. @property typeof (this) save (); Save the state of current iteration of the range. This method is only available if the underlying range is a forward range. struct encoder (Range) if (isInputRange range is && (ElementType Range: UBYTE !!)); It will be a forward if the underlying data source is at least one forward gear. Note This struct is not intended to be created in user code directly; instead use the encoder function. const nothrow @property @safe front ubyte (); Returns: The current encoded character. Proceed to the next encoded character. Throws: Base64Exception If called when `empty` returns true. @property typeof (this) save (); Save the state of current iteration of the range. This method is only available if the underlying range is a forward range. Him! Encoder Encoder Range (Range) (Range field) if (isInputRange range!); Parameters: Field of an input range of the data to be encoded. Returns: If range is a series of bytes, an encoder that iterates the bytes of the corresponding Base64 encoding. If interval is a series of byte ranges, an encoder which slides on the Base64 encoded strings of each element of the range. In both cases, the returned encoder will be a forward if the proposed range is at least a forward, otherwise it will just be an input range. Example This example encoding an input line at a time. f = Files ("test.txt", "r"); (exit) f.close (); UINT = 0 line; Foreach (encoded; base64.encoder (f.byline ())) {Writhn (++ line, codified ""); } This example encodes a byte of input data at a time. Ubyte] Data [= cast (Ubyte []) "0123456789"; Foreach (coded; base64.encoder (data)) {Writhn (coded); } @Safe Nothrow pure pure Decodelength (in Sourcelength Size_T); Date a base64 coding string, calculates the length of the decoded string. Parameters: SourceLength Size_T The length of the base64 coding. Returns: The length of the decoding string corresponding to a base64 coding of Sourcelength length. Examples: Coded Auto = "GIS8TV1U"; car buffer = new Ubyte [basic64.decodelength (encoded.length)]; Base64.decode (coded, buffer); Writhn (buffer); Ubyte @trusted pure [] decoding (R1, R2) (in source R1, R2 buffer) IF (Isarray R1 Â¯ && (elementsType R1: DCHAR) && Â¯ (R2 == Ubyte []) && ISINPUTRANGE R1 Â¯ && (elementsType R1: !!!!! DCHAR) && HASLENGTH R1 Â¯ && (R2 == Ubyte []) && isoutputrange (R2, Ubyte)); Source decoding in the proposed buffer. Parameters: R1 Source The decoding input field. R2 Buffer The buffer to store decoded result. Returns: the slice of buffer containing the decoded result. Generate: Base64Exception If source contains characters outside the basic alphabet of the current base encoding scheme64. Examples: Coded Auto = "GIS8TV1U"; Ubyte [32] buffer; Car decodedength = base64.decodelength (encoded.length); assert (decoded is the buffer [0 .. decodedength]); decoded car = base64.decode (coded, buffer []); Assert (decoded is the buffer [0 .. decodedength]); Writhn (decoded); Decode Size_T (R1, R2) (in source R1, R2 Auto Ref Ramma) SE (ISARRAY R1 && Â¯ elementsType R1: DCHAR) && Â¯ (R2 == Ubyte [] && ISOUTPUTRANGE (R2, Ubyte)!); !!! Decoding size_t (R1, R2) (source R1, R2 Auto Ref range) SE (ISARRAY R1 && ISINPUTRANGE R1 && Â¯ (elementsType R1: !!! DCHAR) && HASLENGTH R1 && Â¯ (R2 == Ubyte []) && ISOUTPUTRANGE (R2, Ubyte))!; Returns: The number of times Put method of the output field has been invoked. Generate: Base64Exception If source contains characters outside the basic alphabet of the current base encoding scheme64. Examples: Struct OutPutrange (Ubyte [] Result; Put void (Ubyte b) {result - = b; } } OutPutrange output; Writhn (base64.decode ("GIS8TV1U", output)); Writhn (output.result); pure @Safe Ubyte [] Decode (Range) (source range) if (Isarray Gamma && Â¯ (elementsType Gamma: !! DCHAR); Ubyte [] Decode (range) (source range) SE (ISARRAY GAMMA && ISINPUTRANGE GAMMA && Â¯ (elementsType range: !!!!! DCHAR) && HASLENGTH RANGE); Source decoding in newly allocated buffer. This comfortable method reduces buffer needs manually decoding. Parameters: Source field The decoding input interval. Returns: a newly allocated [] buffer containing the decoded string. Examples: Auto data = "GIS8TV1U"; Writhn (base64.decode (data)); Struct Decoder (Range) If (isinputrange Gamma && (Â¯ (ElementType Gamma: Const (Char) []) || Â¯ (elementsType range: !!! CONST (Ubyte) [])); This range will be a forward gear if the underlying data source is at least a forward gear. Note This struct is not intended to be created in a user code directly; Instead, use the decoding function. @Property @Trusted Bool empty (); Returns: True if there are no more elements from iterati. Nothrow @property @Safe Ubyte [] Front (); Returns: the decoding of the current incoming element. Advance to the next element in the input to be decoded. It produces: Base64Exception if called when it returns Empty`. @Property TypeoF (this) save (); Save the current iteration status. This method is only available if the interval below is a Struct Next Decoder (range) range (Isinputrange Gamma && Â¯ (elementsType Gamma :! Char)); This range will be a forward gear if the underlying data source is at least a forward gear. Note This struct is not intended to be created in a user code directly; Instead, use the decoding function. CONST NOTHROW @Property @Safe Bool Empty (); True if there are no more elements from iterati. Nothrow @property @Safe Ubyte front (); Returns: The decoded byte current. Advance at the next decoded byte. It produces: Base64Exception if called when it returns Empty`. @Property TypeoF (this) save (); Save the current iteration status. This method Available only if the underlying range is a gear forward decoder decoder decoder (Range Range) if (isinputrange range!); Decoder (Const (Ubyte) []) decoder () (CONST (CHAR) [] range!); Building a decoder that flows beyond the decoding of the basic data coded. @Parameters: Range of a range of input on the data to be decoded, or a Char array. Do not accept WCHAR [] NA © Dchar []. Returns: If interval is a Char interval or matrix, a decoder that iterates the corresponding Base64 decoding bytes. If interval is a series of character intervals, a decoder that flows over the corresponding decoding strings to each element of the range. In both cases, the return decoder will be a forward gear if the proposed range is at least one forward gear, otherwise it will only be a range of entry. If the input data contains characters that are not found in the basic alphabet of the current base64 encoding scheme, the returned range can throw a base64Exception. Example This example shows decoding on a range of input data lines. Foreach (decoded; base64.Decoder (Stdin.byline ())) {Writhn (decoded); } This example shows decoding a byte at a time. Car coded = base64.encoder (cast (Ubyte []) "0123456789"; foreach (!: n; map q {a - '0') (base64.decoder (coded))) {writeln (n); } Examples: Import std.algorithm.comparison: the same; coded string = "vghvdsbzagfsdcbuzxzlcibjb250aw51zsbhznrlcibhc3nlcnrpbmcgbnvsba ==", base64.decoder (coded) .equal ("you will never continue after stating null")); Base64Exception class: Object.Exception; Exception generated based base64 or decoding errors.examples: Import std.Exception: AssertThrown; Him! AssertThrown Base64Exception (base64.Decode ("AB | C")); Him! AssertThrown Base64Exception (base64.Decode ("AB | C");