Natural Functions

This document describes various Software AG Natural functions whose names start with `SAG` and which are based on the principle of a user defined function; see *User-Defined Functions* in the *Programming Guide*.

These Natural functions are delivered in the library `SYSTEM` on system file `FNAT`. Sample function calls are provided in the library `SYSEXPG`.

- URL Encoding
- Base64 Encoding

URL Encoding

Interfacing Natural applications with HTTP requests often requires that the URI (Uniform Resource Identifiers) is URL-encoded. The `REQUEST DOCUMENT` statement needs such a URL to access a document.

URL-Encoding (or Percent-Encoding) is a mechanism to replace some special characters in parts of a URL. Only characters of the US-ASCII character set can be used to form a URL. Some characters of the US-ASCII character set have a special meaning when used in a URL - they are classified as "reserved" control characters, which structure the URL string into different semantic subcomponents. The quasi standard concerning the generic syntax of an URL is laid down in RFC3986, a document composed by the Internet community. It describes under which conditions the URL-Encoding is needed. This includes the representation of characters which are not inside the US-ASCII character set (for example, Euro sign), and it describes the use of reserved characters.

Reserved characters are:

?	=	&	#	!	$	%	’	(	)	*	+	,	/	:	;	@	[	]

Non reserved characters are:

A-Z	a-z	0-9	-	_	.	~

A URL may only consist of reserved and non-reserved characters, other characters are not permitted. If other byte values are needed (which do not correspond to any of the reserved and non-reserved characters) or if reserved characters are used as data (which should not have a special semantic meaning in the URL context), they need to be translated into the "%-encoding" form - a percent sign, immediately followed by the two-digit hexadecimal representation of the code point, due to the Windows-1252 encoding scheme. This causes a plus sign (+) to appear as `%2B`, a percent sign (%) to appear as `%25` and an at sign (@) to appear as `%40` in the string.

The following encoding functions are operating the complete input string. You should take care not to encode a complete URL or parts of it if they contain control characters (reserved characters) which must not be translated into the percent-form. These functions should only be applied to characters not permitted for use in a URL, and to characters with a special meaning inside the URL context, which are supplied as a data item.

Simple Encoding

The single input parameter contains the character string to be encoded or decoded. All data inside is regarded as represented in code page Windows-1252, regardless which session code page is really active at this time. The execution of the `SAGENC`/`SAGDEC` functions does not require Unicode support. The following characters are replaced with the corresponding US-ASCII hexadecimal equivalents.

Character	<	**(**	+	|	**&**	**!**	**$**	*****	**)**	**;**	/	**,**	**%**	>	**?**	**‘**	**:**	**#**	**@**	**’**	=	**"**	^	**[**	**]**	**{**	**}**	\
is encoded to %*nn*	3C	28	2B	7C	26	21	24	2A	29	3B	2F	2C	25	3E	3F	60	3A	23	40	27	3D	22	5E	5B	5D	7B	7D	5C

In addition, a space is replaced with a plus sign (+). All other characters are not translated and remain as they are. The simple encoding function should be sufficient in most cases of URL encoding. The result field returned is of format A dynamic.

The following functions are available:

- `SAGENC` - Simple encoding (format A to format A)
- `SAGDEC` - Simple decoding (format A to format A)

SAGENC - Simple Encoding (Format A to Format A)

The function `SAGENC` encodes a character string into its percent-encoded form. According to standard RFC3986, reserved characters and characters below US-ASCII `x'7F'` (*which are not allowed in a URL*) will be percent-encoded, a space character is replaced with a plus sign (+). Unreserved characters according to RCF3986 and characters above US-ASCII `x'7F'`, such as German umlauts, are not encoded. If you want to encode such characters, use the extended encoding function `SAGENCE`.

`SAGENC`	This is the simple encoding function call.
`SAGENCP`	The copycode containing the prototype definition is used at compilation time only in order to determine the type of the return variable for function call reference and to check the parameters, if this is desired. `SAGENCP` is optional.
`URLX01`	Example program contained in library `SYSEXPG`. `#URL-ENC := SAGENC(<#URL-DEC>)`

SAGDEC - Simple Decoding (Format A to Format A)

The function SAGDEC decodes the percent-encodings as provided by the function SAGENC. Besides the decoding string, no other input parameters are necessary.

SAGDEC	This is the simple decoding function call.
SAGDECP	The copycode containing the prototype definition is used at compilation time only in order to determine the type of the return variable for function call reference and to check the parameters, if this is desired. SAGDECP is optional.
URLX01	Example program contained in library SYSEXPG. `#URL-DEC := SAGDEC(<#URL-ENC>)`

Extended Encoding

The extended function considers all issues which are specified or recommended in RFC 3986. The following parameters may be considered (default settings shown in bold):

1. *<dynamic U-string>* to be encoded/decoded
2. Return code: <> 0 (Natural error) if error in MOVE ENCODED statement.
3. Error character if return code <> 0
4. Space character: %20/+/don’t encode (default: +)
5. Unreserved characters: encode/**don’t encode**
6. Reserved characters: **encode**/don’t encode
7. Other special characters (neither unreserved nor reserved): **encode**/don’t encode
8. Character Percent-Encoding: ISO-8859-1/UTF-8/any other code page/if = ’ ’ then *CODEPAGE (default Natural code page, not default encoding code page!)
9. User-selected character in an X-array of format U, which shall not be percent-encoded according to the above parameters, for example, for the Euro sign character, which is not in the ISO-8859-1 code page, or to prevent a character from percent-encoding.
10. User defined percent-encoding in an X-array of format A, for a user-selected character in the same occurrence of the X-array.

The input parameter for a character string will be in Natural format U. This means the input string may contain all Unicode characters. The output string of the extended function is of format A in the Natural default code page (*CODEPAGE). The code page of the percent-encoding can be selected. The UTF-8, ISO-8859-1, percent-encoding of the Euro sign will be done by the MOVE ENCODED statement. If an input character does not exist in the target code page used for percent-encoding, the character will not be encoded. This means the character will be returned unchanged in the default Natural code page. If the character does not exist in the default Natural code page either, it will be replaced by that substitution character which is returned by the MOVE ENCODED statement. The substitution character will be percent-encoded. This may happen only if the percent-encoding code page is not UTF-8. The last `MOVE ENCODED` error will be returned.

The parameters are optional parameters. If the user does not specify a parameter, the default value will be assumed. If the user specifies an own character translation table, the characters in the table will be percent-encoded according to this table and not according to the other parameters. If the percent-encoding of a character in the user-defined translation table is equal to the character or blank, this character will not be encoded. Thus, single characters from the reserved or unreserved character set can be excluded.

The following functions are available:

- `SAGENCE` - Extended encoding (format U to format A, optional parameters)
- `SAGDECE` - Extended decoding (format A to format U, optional parameters)

SAGENCE - Extended Encoding (Format U to Format A, Optional Parameters)

The function `SAGENCE` percent-encodes a string, using the hexadecimal value of the selected code page (default UTF-8). According to standard RFC3986, reserved characters and characters below US-ASCII `x'7F'`, which are not allowed in a URL, will be percent-encoded. Also, the space and the percent sign (%) will be encoded.

In addition, unreserved characters according to RCF3986 and characters above US-ASCII `x'7F'`, such as German umlauts, will be encoded by this function.

`SAGENCE` needs Natural Unicode support.

SAGENCE	This is the extended encoding function call.
	Parameters:
	```
	P-DEC-STR-E    (U)
	P-RET          (I4)     OPTIONAL  /* 0:    ok
	                                 /* else: Natural error returned
	                                 /*       by the GIVING clause of
	                                 /*       MOVE ENCODED.
	                                 /*       This is the error which
	                                 /*       comes up when a character
	                                 /*       cannot be converted into
	                                 /*       the target code page.
	/* Error strategy:
	/* Step 1: If a character shall be %-encoded and is not available
	/* in the code page for %-encoding, the character will not be
	/* %-encoded. It will be copied.
	/* Step 2: If a character will not be %-encoded but copied from the
	/* input format U-variable to a format A-variable (in *CODEPAGE)
	/* and the character is not available in *CODEPAGE, a substitution
	/* character will be used instead. The substitution character will
	/* be %-encoded.
	/* The last error will be returned in P-RET.
	P-ERR-CHAR     (U1)     OPTIONAL  /* Character causing the error
	P-SPACE        (A1)     OPTIONAL  /* '%'  => %20
	                                 /* ' '  => ' '
	                                 /* else => '+' (default)
	P-UNRES        (A1)     OPTIONAL  /* 'E'  => encode
	                                 /* else => don't encode (default)
	P-RES          (A1)     OPTIONAL  /* 'E'  => encode (default)
	                                /* else => don't encode
	P-OTHER        (A1)     OPTIONAL  /* 'E'  => encode (default)
	                                 /* else => don't encode
	P-CP           (A64)    OPTIONAL  /* IANA name e.g. UTF-8 (default)
	                                 /* or ISO-8859-1
	/* On mainframe only code page names defined with the macro NTCPAGE
	/* in the source module NATCONFG can be used. Other code page names
	/* are rejected with a corresponding runtime error.
	/*
	P-CP-TABLE-CHAR(U1/1:*)  OPTIONAL /* user selected char to be
	                                 /* %-encoded, e.g. 'ö' or '/'
	P-CP-TABLE-ENC (A12/1:*) OPTIONAL /* user %-encoding
	                                 /* e.g. character 'ö'
	                                 /*      '%F6'    -> ISO-8859-1
	                                 /*      '%C3%B6' -> UTF-8
	                                 /* e.g. character '/'
	                                 /*      '/'    -> '/' not encoded
	                                 /*       although P-RES = 'E'
	/* Characters in this table will be encoded according to the
	/* specfied %-encoding. If the U12 encoding part is blank (space
	/* according to *CODEPAGE) or the P-CP-TABLE-ENC value is equal to
	/* the character, then the character will not be encoded at all.
	/*
	```
SAGENCEP	The copycode containing the prototype definition is used at compilation time only in order to determine the type of the return variable for function call reference and to check the parameters, if this is desired. SAGENCEP is optional.
URLX01	Example program contained in library SYSEXPG.

	Sample Calls
	
	Default values will be taken:
	
	`#URL-ENC := SAGENCE(<#URL-DEC-U>)`
	
	All possible parameters are specified:
	
	`#URL-ENC := SAGENCE(<#URL-DEC-U,L-RET,L-ERR-CHAR,L-SPACE,L-UNRES, L-RES,L-OTHER,L-CP,L-CP-TAB-CHAR(*),L-CP-TAB-ENC(*) >)`

SAGDECE - Extended Decoding (Format A to Format U, Optional Parameters)

The function SAGDECE decodes the percent-encodings as provided by the function SAGENCE. If a space character and/or a code page is specified, the values must be the same as specified for encoding.

SAGDECE needs Natural Unicode support.

SAGDECE	This is the extended decoding function call.
	Parameters:

	`1 P-ENC-STR-E    (A)`
	`1 P-RET          (I4)     OPTIONAL  /* 0:    ok`
	`                                     /* else: Natural error returned`
	`                                     /*       by the GIVING clause of`
	`                                     /*       MOVE ENCODED.`
	`                                     /*       This error comes up`
	`                                     /*       when a %-encoded`
	`                                     /*       character cannot be`
	`                                     /*       converted into the`
	`                                     /*       target code page.`
	`  /* The last error will be returned in P-RET.`
	`1 P-ERR-CHAR     (A12)    OPTIONAL  /* Error character %-encoded`
	`1 P-SPACE        (A1)     OPTIONAL  /* ' '  => ' '`
	`                                     /* else => '+' (default)`
	`1 P-CP           (A64)    OPTIONAL  /* IANA name e.g. UTF-8 (default)`
	`                                     /* or ISO-8859-1`
	`/* On mainframe only code page names defined with the macro NTCPAGE`
	`/* in the source module NATCONFG can be used. Other code page names`
	`/* are rejected with a corresponding runtime error.`
	`/*`

SAGDECEP	The copycode containing the prototype definition is used at compilation time only in order to determine the type of the return variable for function call reference and to check the parameters, if this is desired. SAGDECEP is optional.
URLX01	Example program contained in library SYSEXPG.

	Sample Calls
	
	Default values will be taken:
	
	`#URL-DEC-U := SAGDECE(<#URL-ENC>)`
	
	All possible parameters are specified:
	
	`#URL-DEC-U := SAGDECE(<#URL-ENC,L-RET,L-ERR-CHAR-DEC,L-SPACE,L-CP>)`

Example Program

Example program contained in library SYSEXPG:

```
** Example 'URLX01': ENCODED-STR := SAGENC(<DECODED-STR>)
************************************************************************
DEFINE DATA
LOCAL
1 SAMPLE-STRING (A72)
/*
1 #URL-DEC      (A) DYNAMIC
1 #URL-ENC      (A) DYNAMIC
/*
1 #URL-DEC-U    (U) DYNAMIC
/*
1 L-RET         (I4)   /* Return code
1 L-ERR-CHAR    (U1)   /* Error character
1 L-ERR-CHAR-DEC(A12)  /* Decoded error character
```

```
1 L-SPACE       (A1)   /* '%'  => %20, ' '  => ' ',
                       /* else => '+' (default)
1 L-UNRES       (A1)   /* 'E' => encode, else => don't encode (default)
1 L-RES         (A1)   /* 'E' => encode (default), else => don't encode
1 L-OTHER       (A1)   /* 'E' => encode (default), else => don't encode
1 L-CP          (A64)  /* default *CODEPAGE
1 L-CP-TAB-CHAR (U1/1:1)
1 L-CP-TAB-ENC  (A12/1:1)
1 L-MSG         (U72)
END-DEFINE
/*
/*
/*
WRITE 'Sample string to be processed:'
/* The string below shall be encoded and decoded again.
/* After decoding it should be unchanged.
SAMPLE-STRING := '"Decoded data!"'
WRITE SAMPLE-STRING (AL=72) /
/*
/* Assign the sample string to the input variable #URL-DEC of the
/* simple encoding function.
#URL-DEC      := SAMPLE-STRING
/*
/* Copycode SAGENCP containing the prototype definition is used at
/* compilation time only in order to determine the type of the return
/* variable for function call reference and to check the parameters,
/* if this is desired. SAGENCP is optional.
INCLUDE SAGENCP
/*
/* SAGENC(<#URL-DEC>) is the simple encoding function call.
/*
/* Function SAGENC %-encodes a string to code page ISO-8859-1.
/* According to standard RFC3986 reserved characters and characters
/* below US-ASCII x'7F' which are not allowed in a URL will be
/* %-encoded.
/* Also the space and the percent sign will be encoded.
/* Unreserved characters according to RCF3986 and characters above
/* US-ASCII x'7F' will not be encoded. If you want to encode such
/* characters, use the extended encoding function.
/*
/* ---- Space                       ' ' -> '+'
/* ---- Percent sign                '%' -> '%25'
/*
/* Unreserved characters according to RFC3986 (will not be encoded!):
/* ---- Period (fullstop)           '.' -- '%2E'
/* ---- Tilde                       '~' -- '%7E'
/* ---- Hyphen                      '-' -- '%2D'
/* ---- Underscore character        '_' -- '%5F'
/* ---- digits, lower and upper case characters
/* ---- 0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
/*
/* Reserved characters according to RFC3986:
/* ---- Exclamation mark            '!' -> '%21'
/* ---- Number sign                 '#' -> '%23'
/* ---- Dollar sign                 '$' -> '%24'
/* ---- Ampersand                   '&' -> '%26'
/* ---- Apostrophe                  ''' -> '%27'
/* ---- Left parenthesis            '(' -> '%28'
/* ---- Right parenthesis           ')' -> '%29'
/* ---- Asterisk                    '*' -> '%2A'
/* ---- Plus sign                   '+' -> '%2B'
/* ---- Comma                       ',' -> '%2C'
/* ---- Reverse solidus (backslash) '/' -> '%2F'
/* ---- Colon                        ':' -> '%3A'
/* ---- Semi-colon                   ';' -> '%3B'
/* ---- Equals sign                  '=' -> '%3D'
/* ---- Question mark                '?' -> '%3F'
/* ---- Commercial at                '@' -> '%40'
/* ---- Square bracket open          '[' -> '%5B'
/* ---- Square bracket close         ']' -> '%5D'
/*
/* Other characters below x'7F' (US-ASCII) but not allowed in URL
/* ---- Quotation mark               '"' -> '%22'
/* ---- Less than                    '<' -> '%3C'
/* ---- Greater than                 '>' -> '%3E'
/* ---- Reverse solidus (backslash) '\' -> '%5C'
/* ---- Accent, Circumflex           '^' -> '%5E'
/* ---- Accent, Grave                ''' -> '%60'
/* ---- Opening brace                '{' -> '%7B'
/* ---- Vertical bar                 '|' -> '%7C'
/* ---- Closing brace                '}' -> '%7D'
/*
#URL-ENC := SAGENC(<#URL-DEC>)
/*
/*
WRITE 'Simple function, encoded:'
WRITE #URL-ENC (AL=72)
/*
/* Copycode SAGDECP containing the prototype definition is used at
/* compilation time only in order to determine the type of the return
/* variable for function call reference and to check the parameters,
/* if this is desired. SAGDECP is optional.
INCLUDE SAGDECP
/*
/* SAGDEC(<#URL-ENC>) is the simple decoding function call.
/* It decodes the above described %-encodings.
/*
#URL-DEC := SAGDEC(<#URL-ENC>)
/*
/*
/* The result after encoding and decoding must be equal to the original
/* SAMPLE-STRING.
WRITE 'Simple function, decoded:'
WRITE #URL-DEC (AL=72)
/*
/*
/*
WRITE /
/*
/*
/*
/* Assign the sample string to the input variable #URL-DEC-U of the
/* enhanced encoding function.
#URL-DEC-U := SAMPLE-STRING
/*
/* Copycode SAGENCEP containing the prototype definition is used at
/* compilation time only in order to determine the type of the return
/* variable for function call reference and to check the parameters,
/* if this is desired. SAGENCEP is optional.
INCLUDE SAGENCEP
/*
/* This is the enhanced encoding function call.
/* The way, characters will be %-encoded dependes on the input
/* parameter of the function.
/* The parameters of the encoding and decoding function are preset
/* with the default values.
/* L-CP-TAB-CHAR(*) and L-CP-TAB-ENC(*) don’t have default values.
/* L-CP-TAB-CHAR(1) = ’ä’ and L-CP-TAB-ENC(1) = ’%C3%A4’ will not be
/* used for the sample string ’"Decoded data!"’. The string does not
/* contain an ’ä’.
L-SPACE     := ’+’          /* encoding and decoding
L-UNRES     := ’D’          /* encoding only
L-RES       := ’E’          /* encoding only
L-OTHER     := ’E’          /* encoding only
L-CP        := ’UTF-8’      /* encoding and decoding
                            /* e.g. ISO-8859-1, UTF-16BE, UTF-32BE
L-CP-TAB-CHAR(1) := ’ä’     /* encoding only
L-CP-TAB-ENC (1) := ’%C3%A4’ /* encoding only
/*
/* Note that all possible parameters are specified for this sample
/* call.
/* If the default values shall be used and no return code is wanted,
/* all parameters can be omitted, besides the string #URL-DEC-U.
/*
#URL-ENC := SAGENCE(<#URL-DEC-U,L-RET,L-ERR-CHAR,L-SPACE,L-UNRES,
  L-RES,L-OTHER,L-CP,L-CP-TAB-CHAR(*),L-CP-TAB-ENC(*) >)
WRITE ’Extended function, encoded:’
WRITE #URL-ENC (AL=72)
IF L-RET NE 0 THEN
  /* If L-RET = 0, the function worked ok. Else L-RET contains the
  /* Natural error returned by the GIVING clause of MOVE ENCODED.
  /* The error comes up when a character cannot be converted into
  /* the target codepage, e.g. because a character does not exist
  /* in the target codepage.
  COMPRESS ’Error’ L-RET ’with MOVE ENCODED of’ L-ERR-CHAR INTO L-MSG
  WRITE L-MSG
END-IF
/*
/* Copycode SAGDECEP containing the prototype definition is used at
/* compilation time only in order to determine the type of the return
/* variable for function call reference and to check the parameters,
/* if this is desired. SAGDECEP is optional.
INCLUDE SAGDECEP
/*
/* This is the 1st enhanced decoding function call with 5 parameters.
/* Note that all possible parameters are specified for this sample
/* call.
/* Since the parameters have the default values, the subsequent
/* function calls return the same result although parameters
/* have been omitted.
#URL-DEC-U := SAGDECE(<#URL-ENC,L-RET,L-ERR-CHAR-DEC,L-SPACE,L-CP>)
WRITE ’Extended function, decoded:’
WRITE #URL-DEC-U (AL=72)
IF L-RET NE 0 THEN
  /* If L-RET = 0, the function worked ok. Else L-RET contains the
  /* Natural error returned by the GIVING clause of MOVE ENCODED.
  /* The error comes up when a %-encoded character cannot be converted
  /* into the target codepage, e.g. because a character does not exist
  /* in the target codepage.
  COMPRESS ’Error’ L-RET ’with MOVE ENCODED of’ L-ERR-CHAR INTO L-MSG
  WRITE L-MSG
  RESET L-RET
END-IF
/*
/* This is the 2nd enhanced decoding function call with one parameter.
#URL-DEC-U := SAGDECE(<#URL-ENC>)
```

```
WRITE #URL-DEC-U (AL=72)
/* L-RET will not be returned
/*
/* This is the 3rd enhanced decoding function call with 3 parameters.
#URL-DEC-U := SAGDECE(<#URL-ENC,L-RET,2X,L-CP>)
WRITE #URL-DEC-U (AL=72)
IF L-RET NE 0 THEN
  COMPRESS 'Error' L-RET 'with MOVE ENCODED of' L-ERR-CHAR INTO L-MSG
  WRITE L-MSG
  RESET L-RET
END-IF
/*
END
```

Base64 Encoding

This section describes Natural functions which can be used to convert binary data into printable, network-compatible data or vice versa, using Base64 conversion.

Base64 conversion means conversion from format B to format A and back to format B, where 6 (binary) bits will be converted into 8 (alphanumerical) bits; for example, a B3 value will be converted into an A4 value.

Anmerkung:
Every binary value will be converted into a non-ambiguous alphanumerical value. Re-converting this alphanumerical value again will result in the original binary value. However, this is not the case for most of the format A to format B and back to format A conversions.

The conversion may be used to transfer a `.bmp` file via TCP/IP, or to transfer Natural binary or integer values via the utility protocol.

On Open Systems only: There are 3 modes available: `RFC3548`, `RFC2045` and `NATRPC` (default). `NATRPC` means the conversion is done according the `NATRPC` logic. This is 100% mainframe compatible. `RFC2045` is the default of the CMBASE64 call. `RFC3548` is like `NATRPC`, but alphanumerical bytes which are not needed are filled with an equals sign character (=).

The following functions are available:

- `SAG64BA` - Binary to Alphanumerical Conversion
- `SAG64AB` - Alphanumerical to Binary Conversion

These two functions together provide the same functionality as the Natural application programming interface `USR4210N`, which is delivered in library `SYSEXT`.

SAG64BA - Binary to Alphanumerical Conversion

The function `SAG64BA` converts binary data into printable, network-compatible data, using Base64 encoding.

SAG64BA	This is the binary to alphanumerical format conversion function.
	Parameters:

	`1 PARM-B         (B)      DYNAMIC BY VALUE`
	`                 /* Binary source input/target output`
	`1 PARM-RC        (I4)     OPTIONAL`
	`                 /* 0:    ok`
	`                 /* Mainframe`
	`                 /* 1  Source is not numeric`
	`                 /* 2  Source is not packed`
	`                 /* 3  Source is not floating point`
	`                 /* 4  Overflow, source doesn’t fit into target`
	`                 /* 5  Integer overflow`
	`                 /* 6  Source is not a valid date or time`
	`                 /* 7  Length error (hex input not even)`
	`                 /* 8  Target precision is less than source precision`
	`                 /* 9  Float underflow (result->0)`
	`                 /* 10 Alpha source contains non-hex characters`
	`                 /* 20 Invalid function code`
	`                 /* Open Systems`
	`                 /* 1  Invalid value for RFC parameter`
	`                 /* 2  Invalid function code`
	`                 /* 3  CMBASE64: Overflow, source doesn’t fit into`
	`                 /*              target`
	`                 /* 4  CMBASE64: Non-base64 character found in encoded`
	`                 /*              data`
	`                 /* 5  CMBASE64: Out of memory`
	`                 /* 6  CMBASE64: Invalid number of parameters`
	`                 /* 7  CMBASE64: Invalid parameter type`
	`                 /* 8  CMBASE64: Invalid parameter length`
	`                 /* 9  CMBASE64: Invalid function code`
	`                 /* 10 CMBASE64: Unkown return code`
	`1 PARM-ERRTXT    (A72)    OPTIONAL`
	`                 /* blank, if ok no error`
	`                 /* else error text`
	`1 PARM-RFC       (B1)     OPTIONAL`
	`                 /* OS only, not used for MF`
	`                 /* 0 - RFC3548; 3 - RFC2045; 4 - NATRPC;`

SAG64BAP	The copycode containing the prototype definition is used at compilation time only in order to determine the type of the return variable for function call reference and to check the parameters, if this is desired. SAG64BAP is optional.

B64X01	Example program contained in library SYSEXPG.
	
	Default values will be taken:
	
	`PARM-A := SAG64BA(<PARM-B>)`
	
	All possible parameters are specified (PARM-RFC does not apply to mainframe):
	
	`PARM-A := SAG64BA(<PARM-B,PARM-RC,PARM-ERRTXT,PARM-RFC>)`

SAG64AB - Alphanumerical to Binary Conversion

The function SAG64AB converts printable, network-compatible data into binary data, using Base64 encoding.

SAG64AB	This is the alphanumerical to binary format conversion function.
	Parameters:
	```
	1 PARM-A         (A)
	                /* Alpha source input/target output
	1 PARM-RC        (I4)     OPTIONAL
	                /* 0:    ok
	                /* Mainframe
	                /* 1  Source is not numeric
	                /* 2  Source is not packed
	                /* 3  Source is not floating point
	                /* 4  Overflow, source doesn’t fit into target
	                /* 5  Integer overflow
	                /* 6  Source is not a valid date or time
	                /* 7  Length error (hex input not even)
	                /* 8  Target precision is less than source precision
	                /* 9  Float underflow (result->0)
	                /* 10 Alpha source contains non-hex characters
	                /* 20 Invalid function code
	                /* Open Systems
	                /* 1  Invalid value for RFC parameter
	                /* 2  Invalid function code
	                /* 3  CMBASE64: Overflow, source doesn’t fit into
	                /*             target
	                /* 4  CMBASE64: Non-base64 character found in encoded
	                /*             data
	                /* 5  CMBASE64: Out of memory
	                /* 6  CMBASE64: Invalid number of parameters
	                /* 7  CMBASE64: Invalid parameter type
	                /* 8  CMBASE64: Invalid parameter length
	                /* 9  CMBASE64: Invalid function code
	                /* 10 CMBASE64: Unkown return code
	1 PARM-ERRTXT    (A72)    OPTIONAL
	                /* blank, if ok no error
	                /* else error text
	1 PARM-RFC       (B1)     OPTIONAL
	                /* OS only, not used for MF
	                /* 0 - RFC3548; 3 - RFC2045; 4 - NATRPC;
	```
SAG64ABP	The copycode containing the prototype definition is used at compilation time only in order to determine the type of the return variable for function call reference and to check the parameters, if this is desired. SAG64ABP is optional.

B64X01	Example program contained in library SYSEXPG.
	
	Default values will be taken:
	
	`PARM-B := SAG64AB(<PARM-A>)`
	
	All possible parameters are specified (PARM-RFC does not apply to mainframe):
	
	`PARM-B := SAG64AB(<PARM-A,PARM-RC,PARM-ERRTXT,PARM-RFC>)`

Example Program

Example program `B64X01` contained in library `SYSEXPG`:

```
** Example ’B64X01’: BASE64-A-STR := SAG64BA(<BASE64-B-STR>)
************************************************************************
* Function ......... Convert binary data into printable,
*                    network-compatible data or vice versa using
*                    Base64 encoding.
*
*                    Base64 encoding means (B) -> (A) -> (B),
*                    where 6 (binary) bits will be encoded into 8
*                    (alpha) bits, e.g a (B3) value will be encoded
*                    into a (A4) value.
*
*                    Note: Every binary value will be encoded into
*                    a non-ambiguous alpha value. Re-encoding this
*                    alpha value again will result in the original
*                    binary value. However, this is not the case with
*                    most of the (A) -> (B) -> (A) encodings.
*
*                    The encoding may be used to transfer a .bmp
*                    file via TCP/IP, or to transfer Natural binary or
*                    integer values via the utility protocol.
*
*                    Open Systems only:
*                    On Open Systems, there are 3 modes:
*                    RFC3548, RFC2045 and NATRPC (default).
*                    NATRPC means the encoding follows
*                    the NATRPC logic. This is 100% MF compatible.
*                    RFC2045 is the default of the CMBASE64 call.
*                    RFC3548 is like NATRPC, but alpha bytes not
*                    needed are filled with ’=’.
*
DEFINE DATA
LOCAL
1 FUNCTION        (A2)
                  /* ’AB’ Alpha to binary encoding
                  /* ’BA’ Binary to alpha encoding
1 PARM-RC         (I4)
                  /* 0:    ok
                  /* Mainframe
                  /* 1  Source is not numeric
                  /* 2  Source is not packed
                  /* 3  Source is not floating point
                  /* 4  Overflow, source doesn’t fit into target
                  /* 5  Integer overflow
                  /* 6  Source is not a valid date or time
                  /* 7  Length error (hex input not even)
                  /* 8  Target precision is less than source precision
                  /* 9  Float underflow (result->0)
                  /* 10 Alpha source contains non-hex characters
                  /* 20 Invalid function code
                  /* Open Systems
                  /* 1  Invalid value for RFC parameter
                  /* 2  Invalid function code
                  /* 3  CMBASE64: Overflow, source doesn’t fit into
                  /*              target
                  /* 4  CMBASE64: Non-base64 character found in encoded
                  /*              data
                  /* 5  CMBASE64: Out of memory
                  /* 6  CMBASE64: Inalid number of parameters
                  /* 7  CMBASE64: Invalid parameter type
                  /* 8  CMBASE64: Invalid parameter length
                  /* 9  CMBASE64: Invalid function code
                  /* 10 CMBASE64: Unkown return code
1 PARM-ERRTXT     (A72)
                  /* blank, if ok no error
                  /* else error text
1 PARM-A          (A)   DYNAMIC
                  /* Alpha source input/target output
1 PARM-B          (B)   DYNAMIC
*                 /* Binary source input/target output
1 PARM-RFC        (B1)
                  /* OS only, not used for MF
                  /* 0 - RFC3548; 3 - RFC2045; 4 - NATRPC;
/*
1 #BACKUP-A       (A) DYNAMIC
1 #BACKUP-B       (B) DYNAMIC
END-DEFINE
/*
/*
SET KEY ALL
/*
/* Copycode SAG64BAP and SAG64ABP containing the prototype definition
/* is used at compilation time only in order to determine the type of
/* the return variable for function call reference and to check the
/* parameters, if this is desired. SAG64BAP and SAG64ABP are optional.
INCLUDE SAG64BAP
INCLUDE SAG64ABP
/*
REPEAT
  RESET PARM-A PARM-B
  REDUCE DYNAMIC PARM-A TO 0
  REDUCE DYNAMIC PARM-B TO 0
  FUNCTION := 'BA'
  PARM-B := H'0123456789ABCDEF'
  INPUT (AD=MIL IP=OFF CD=NE) WITH TEXT PARM-ERRTXT
   // 10T 'Base64 Encoding:' (YEI)
    / 10T '-' (19) (YEI) /
    / 10T 'Function (BA,AB) ..' (TU) FUNCTION (AD=T)
    / 10T 'Alpha In/Output ...' (TU) PARM-A (AL=30)
    / 10T 'Binary In/Output ..' (TU) PARM-B (EM=HHHHHHHH)
    / 10T 'Response ..........' (TU) PARM-RC (AD=OD CD=TU)
    / PARM-ERRTXT (AD=OD CD=TU)
  RESET PARM-ERRTXT
  IF *PF-KEY NE 'ENTR'
    ESCAPE BOTTOM
  END-IF
  /*
  RESET #BACKUP-A #BACKUP-B
  REDUCE DYNAMIC #BACKUP-A TO 0
  REDUCE DYNAMIC #BACKUP-B TO 0
  #BACKUP-A := PARM-A
  #BACKUP-B := PARM-B
  /*
  IF FUNCTION = 'BA'
    /* Parameter PARM-RC, PARM-ERRTXT and PARM-RFC are optional
    /* Parameter PARM-RFC does not apply to mainframe
    /* PARM-A := SAG64BA(<PARM-B,PARM-RC,PARM-ERRTXT,PARM-RFC>)
    PARM-A := SAG64BA(<PARM-B,PARM-RC,PARM-ERRTXT>)
    /* PARM-A := SAG64BA(<PARM-B,PARM-RC>)
    /* PARM-A := SAG64BA(<PARM-B>)
  ELSE
```

```
    /* Parameter PARM-RC, PARM-ERRTXT and PARM-RFC are optional
    /* Parameter PARM-RFC does not apply to mainframe
    /* PARM-B := SAG64AB(<PARM-A,PARM-RC,PARM-ERRTXT,PARM-RFC>)
    PARM-B := SAG64AB(<PARM-A,PARM-RC,PARM-ERRTXT>)
    /* PARM-B := SAG64AB(<PARM-A,PARM-RC>)
    /* PARM-B := SAG64AB(<PARM-A>)
  END-IF
  /*
  IF PARM-RC NE 0 THEN
    WRITE 'Encoding' FUNCTION
    WRITE NOTITLE PARM-ERRTXT
  ELSE
    IF FUNCTION = 'BA' THEN
      WRITE 'Binary -> Alpha'
      WRITE '=' PARM-B (EM=HHHHHHHHHHHHHHHHHHHHHHHHHH)
        / '=' PARM-A (AL=50)
      RESET PARM-B
      REDUCE DYNAMIC PARM-B TO 0
      FUNCTION := 'AB'
    ELSE
      WRITE 'Alpha -> Binary'
      WRITE '=' PARM-A (AL=50) /
        '=' PARM-B (EM=HHHHHHHHHHHHHHHHHHHHHHHHHH)
      RESET PARM-A
      REDUCE DYNAMIC PARM-A TO 0
      FUNCTION := 'BA'
    END-IF
    /*
    IF FUNCTION = 'BA'
      /* Parameter PARM-RC, PARM-ERRTXT and PARM-RFC are optional
      /* Parameter PARM-RFC does not apply to mainframe
      /* PARM-A := SAG64BA(<PARM-B,PARM-RC,PARM-ERRTXT,PARM-RFC>)
      PARM-A := SAG64BA(<PARM-B,PARM-RC,PARM-ERRTXT>)
      /* PARM-A := SAG64BA(<PARM-B,PARM-RC>)
      /* PARM-A := SAG64BA(<PARM-B>)
    ELSE
      /* Parameter PARM-RC, PARM-ERRTXT and PARM-RFC are optional
      /* Parameter PARM-RFC does not apply to mainframe
      /* PARM-B := SAG64AB(<PARM-A,PARM-RC,PARM-ERRTXT,PARM-RFC>)
      PARM-B := SAG64AB(<PARM-A,PARM-RC,PARM-ERRTXT>)
      /* PARM-B := SAG64AB(<PARM-A,PARM-RC>)
      /* PARM-B := SAG64AB(<PARM-A>)
    END-IF
    IF PARM-RC NE 0 THEN
      WRITE 'Encoding' FUNCTION
      WRITE NOTITLE PARM-ERRTXT
    ELSE
      IF FUNCTION = 'BA' THEN
        WRITE 'Binary -> Alpha'
        WRITE '=' PARM-B (EM=HHHHHHHHHHHHHHHHHHHHHHHHHH)
          / '=' PARM-A (AL=50)
        IF PARM-A = #BACKUP-A THEN
          WRITE '******** Encoding successful ********'
        ELSE
          WRITE '******** Value changed by encoding ********'
        END-IF
      ELSE
        WRITE 'Alpha -> Binary'
        WRITE '=' PARM-A (AL=50) /
          '=' PARM-B (EM=HHHHHHHHHHHHHHHHHHHHHHHHHH)
        IF PARM-B = #BACKUP-B THEN
          WRITE '******** Encoding successful ********'
        ELSE
          WRITE '******** Value changed by encoding ********'
        END-IF
      END-IF
    END-IF
  END-IF
END-REPEAT
END
```