Field Codes Reference Guide

2/16/2025

Table of Contents

Search the table of all articles in this guide, listed in alphabetical order, to find the article you need.

Related documentation:

-

Get started

Contents

- 
- Administrator

Learn about the topics covered in the Field Codes Reference Guide.

Related documentation:

-

This guide describes Field_Code that are used in Standard_Response. For more information, see [[PEC-ROU/Current/Designer/FieldCodes|]] and [[PEC-ROU/Current/Designer/StandardResponses|]].

With field codes, you can compose standard responses that are automatically personalized when they are used. This feature is very similar to the Mail-Merge feature in word-processing applications such as Microsoft Word. Consider, for example, this standard response:

```
Dear ,

...
```

This response has two field codes. When an agent inserts this response into an email, the first field code, , is replaced by the contact's first name as it appears in Universal Contact Server. The second field code, , is replaced by the agent's signature as it appears in Configuration Manager.

For example, if an agent named Danielle uses this standard response while replying to an email from a contact named Sam, the result might look like this:

```
Dear Sam,

...

Thank you for choosing My Cloud Security Systems.

Sincerely,
Danielle Rodriguez
Customer Support
www.MCSS.com
```

Topics in this guide

The basics

- Data types
- Operator precedence

- Named constants

Functions

- String functions
- Date and time functions
- Type conversion
- Mathematical functions
- Miscellaneous functions

Objects

- Agent object
- Contact object
- Interaction object

Data types

Contents

- 

- Administrator

Learn about the data types you can use in field codes.

Related documentation:

-

Field codes can use the data types detailed on this page.

Number

You use numbers in field code formulas in much the same way you would in other applications, such as Microsoft Excel. All arithmetic calculations are performed internally using floating point arithmetic (with the decimal point). Rounding occurs only during formatting.

When you write numbers in formulas, you can use scientific notation (for example, `12.34e-2` is the same as `0.1234`).

The Operators table lists the operators that you can use with numbers. Some rows show more than one symbol for the same operator. In these cases, the symbols are synonyms.

Operators

Operator	Description	Example	Result
-	Unary Minus	-4	-4
^	Exponentiation	2^3	8
*	Multiplication	2*3	6
/	Division	8/2	4
Mod	Modulus (Remainder)	14 Mod 5	4
+	Addition	2 + 3	5
-	Subtraction	2 – 3	-1
> GT	Greater Than	2 > 3	False
>= GE	Greater Than or Equal To	2 >= 2	True
	Less Than	2	True
	Less Than or Equal To	2	True
= == EQ	Equal To	2 = 3	False
!= NE	Not Equal To	2 3	True

Operator	Description	Example	Result
:	Format	2 : “#.##”	2.00

String

Use the String data type to represent textual data. When you write a string in a formula, you must enclose it in double quotation marks. For example:

"The sixth sheik's sixth sheep's sick."

You can use the escape sequences shown in the Escape Sequences table to include special characters in a string, such as tabs or carriage returns.

It is also possible to use HTML tags in field codes.

Escape Sequences

Escape	Translates to
\a	Alert (Bell)
\b	Backspace
\f	Form Feed
\n	Line Feed (Newline)
\r	Carriage Return
\t	Horizontal Tab
\v	Vertical Tab
\'	Single Quotation Mark
\"	Double Quotation Mark
\\	Backslash

The Operators and Strings table lists the operators that you can use with strings. All the comparison operators are case insensitive. Some rows show more than one symbol for the same operator. In these cases, the symbols are synonyms.

Operators and Strings

Symbol	Meaning	Example	Result
+	Concatenation	"How" + "die"	"Howdie"
>GT	Greater Than	"A" > "B"	False
>=GE	Greater Than or Equal To	"A" >= "B"	False
	Less Than	"A"	True
	Less Than or Equal To	"A"	True
= ==EQ	Equal To	"A" = "a"	True
!= NE	Not Equal To	"A" NE "B"	True

Date and time

Date/Time values in field code formulas represent specific moments (for example, February 3, 2002, at 10:03:55 AM). The most common operations performed on Date/Times are comparisons (for example,

If you subtract two Date/Time values, the result is the number of days between them. See the Date/Time Example 1 table for examples.

Date/Time Example 1

Formula	Result
Date(2002, 11, 23) – Date(2002, 11, 22)	1
Date(2002, 11, 22) – Date(2002, 11, 23)	-1
Date(2002, 11, 23) – Date(2002, 11, 23, 12)	-0.5

If you add (or subtract) a number to (from) a Date/Time, the result is the Date/Time moved forward (or backward) by that many days. See the Date/Time Example 2 table for examples.

Date/Time Example 2

Formula	Result
Date(2003, 11, 23) + 1	2003-11-24 00:00:00
Date(2003, 11, 23) – 0.5	2003-11-22 12:00:00

Boolean

Set Boolean values in field code formulas to either `True` or `False`. You can use the `True` and `False` keywords to write a Boolean value explicitly, although this is rarely required. Comparison operators (for example, `and so on) always yield Boolean results.`

The Operators and Booleans table lists the operators that you can use with Booleans. Some rows show more than one symbol for the same operator. In these cases, the symbols are synonyms.

Operators and Booleans

Symbol	Meaning	Example	Result
Not !	Unary Not	Not False Not True	True False
And &&	Logical And	False And False False And True True And False True And True	False False False True
Or {{!}}{{!}}	Logical Or	False Or False False Or True True Or False True Or True	False True True True
XOr	Logical Exclusive Or	False XOr False False XOr True True XOr False True XOr True	False True True False
= == EQ	Equal To	True = False	False

Symbol	Meaning	Example	Result
!= NE	Not Equal To	True False	True

Operator precedence

- 
- Administrator

Learn about operator precedence in field codes.

Related documentation:

-

The Operator Precedence table lists all the operators that you can use in field-code formulas.

- Unary operators are shown with `[Unary]` after their symbols.
- The operators are listed in order of precedence, with operators of higher precedence above those of lower precedence.
- Operators in the same row have the same precedence. If two operators of the same precedence are used in a formula, then they are computed left to right if they are binary, and right to left if they are unary.
- You can write some operators using more than one symbol. In these cases, the alternatives are shown in parentheses.

Operator Precedence Table

Operator
+ [Unary], - [Unary]
^
*, /, Mod
+, -
(GT), >= (GE), = (==, EQ), (!=, NE)
Not (!) [Unary]
And (&&)
XOr
Or ({{!}}{{!}})
:

Named constants

- 
- Administrator

Learn about named constants in field codes.

Related documentation:

-

The Keyword Equivalents table lists keywords that are equivalent to certain useful values. Many of these values can be represented in other ways, but the keywords are provided for convenience.

Keyword Equivalents Table

Keyword	Equivalent
iccCr	"\r"
iccLf	"\n"
iccCrLf	"\r\n"
iccBackslash	"\\"
Null	None
True	None
False	None
Pi	3.14159265358979
E	2.71828182845904

String functions

Contents

- 
- Administrator

Learn about the string functions you can use in field codes.

Related documentation:

-

Field codes can use the string functions detailed on this page.

Find

Find(*SearchIn, SearchFor*)

Finds a substring within a string. Returns the `0`-based character position of the found substring. Returns –1 if the substring is not found.

Argument	Description
SearchIn	The string to search in.
SearchFor	The string to search for.

Examples of Find String

Example	Result
	0
	3
	-1

Left

Left(*String, Number*)

Returns a string containing a specified number of characters from the left side of a specified string.

Argument	Description
String	The string from which the leftmost characters are returned.

Argument	Description
Number	The number of characters to return. If 0, an empty string ("") is returned. If greater than the length of String, then the entire string is returned.

Examples of Left String

Example	Result
	"Hello"
	""
	"Hello, World!"

Length

Length(*String*)

Returns the length of a string.

Example of Length String

Example	Result
	5

Mid

Mid(*String, Start, Length*)

Returns a specified substring of a string.

Argument	Description
String	The string from which the substring is returned.
Start	The 0-based character position at which the substring begins. If Start is greater than the length of String, then an empty string ("") is returned.
Length	The number of characters to return. If Length is 0, then an empty string ("") is returned. If Length is greater than the portion of String after Start, then all the characters after Start are returned.

Examples of Mid String

Example	Result
	`"llo"`
	`""`

Example	Result
	`"World!"`

Replace

Replace(*String, Find, ReplaceWith*)

Returns a string in which all instances of a specified substring have been replaced with another string.

Argument	Description
String	The string containing the substring to replace
Find	The substring to search for
ReplaceWith	The replacement string

Examples of Replace String

Example	Result
	"He**o"
	"Hello"
	""

Right

Right(*String, Number*)

Returns a string containing a specified number of characters from the right side of a specified string.

Argument	Description
String	The string from which the rightmost characters are returned.
Number	The number of characters to return. If 0, an empty string ("") is returned. If greater than the length of String, then the entire string is returned.

Examples of Right String

Example	Result
	"orld!"
	""
	"Hello, World!"

ToLower

ToLower(*String*)

Returns a string that has been converted to lowercase.

Example of ToLower String

Example	Result
	"hello, world!"

ToUpper

ToUpper(*String*)

Returns a string that has been converted to uppercase.

Example of ToUpper String

Example	Result
	"HELLO, WORLD!"

Trim

Trim(*String*, [*CharSet*])

Returns a copy of a specified string without specified leading or trailing characters.

Argument	Description
String	The string from which to trim
CharSet	Optional. The characters to trim. If omitted, then white space (" \t\r\n") is trimmed.

Examples of Trim String

Example	Result
	"Howdie"
	"ie"
	"Howd"

TrimLeft

TrimLeft(*String*, [*CharSet*])

The same as `Trim`, except it trims only leading characters.

TrimRight

TrimRight(*String*, [*CharSet*])

The same as `Trim`, except it trims only trailing characters.

Wrap

Trim(*String*, *LineLength*, [*LinePrefix*, [*Eol*]])

Returns a string that has been word-wrapped to a specified line length.

Argument	Description
String	The string to wrap.
LineLength	The maximum length, in characters, of any line, including `LinePrefix` (if specified), but not `Eol`.
LinePrefix	Optional. A string to prefix to each line. Often used to “quote” e-mails being replied to. If omitted, lines are not prefixed.
Eol	Optional. A string to use as a line terminator. If omitted, lines are terminated with `"\r\n"` as usual.

Example: `"Once upon a midnight dreary",",`

Result: `>Once upon*a midnight*dreary*`

Date and time functions

Contents

- 
- Administrator

Learn about the date and time functions you can use in field codes.

Related documentation:

-

Field codes can use the date/time functions detailed on this page.

Date

Date(*Year, Month, Day* [, *Hour[, Minute[, Second*]]]) Or Date(String[, String])

Returns a `Date/Time` constructed from individual components or a string.

> **Important**
> Date(String[, String]) is not recommended.

When using the first syntax function, the optional arguments each default to `0` if omitted. For example, is equivalent to .

When using the second syntax function, the date is constructed by parsing the first string. If the optional argument is omitted, the value of an internal configuration option is used, if present - contact your Genesys representative for details. Otherwise, the platform locale is used. For example:

- if the `fieldcode-format-locale` option or platform locale is set to `en_US`.
-

> **Important**
> Avoid using this second syntax function, since it successively tries multiple Date/Time patterns in order to parse the first argument and so consumes a great deal of CPU time. Also, these patterns are not very lenient. For example, will not parse due to the word at. This method of constructing Date/Time values is less exact than specifying the individual components directly, and may yield incorrect results if the day appears before the month.

Date String

Argument	Description
First argument	The string to parse.
Second argument	Optional. The locale that must be used to parse the first segment. Some examples include: en_US for English (United States), en_GB for English (United Kingdom), and fr_FR for French (France). The value must be in the following format: _

Day

Day(*DateTime*)

Returns the numeric day component of a `Date/Time (1 to 31).`

Hour12

Hour12(*DateTime*)

Returns the numeric hour component of a `Date/Time` based on a 12-hour clock `(1 to 12).`

Hour24

Hour24(*DateTime*)

Returns the numeric hour component of a `Date/Time` based on a 24-hour clock `(0 to 23).`

IsAm

IsAm(*DateTime*)

Returns a Boolean indicating whether a specified `Date/Time` is AM (between midnight and noon). `True` indicates AM and `False` indicates PM.

IsPm

IsPm(*DateTime*)

Returns a Boolean indicating whether a specified `Date/Time` is PM (between noon and midnight).

`True` indicates PM and `False` indicates AM.

Minute

Minute(*DateTime*)

Returns the numeric minute component of a `Date/Time (0–59)`.

Month

Month(*DateTime*)

Returns the numeric month component of a `Date/Time (1–12)`.

MonthName

MonthName*(Arg[, String])*

Converts a month number or a Date/Time to a month name. If the optional argument is omitted, the value of an internal configuration option is used, if present - contact your Genesys representative for details. Otherwise, the platform locale is used.

MonthName String

Argument	Description
First argument	If it is a numeric value (1 to 12), it is converted to the appropriate month name. If it is a Date/Time, the month number is extracted and converted.
Second argument	Optional. The locale that must be used to format the first argument. Some examples include: en_US for English (United States), en_GB for English (United Kingdom), and fr_FR for French (France). The value must be in the following format: _

MonthNameShort

MonthNameShort*(Arg[, String])*

The same as the MonthName, but this returns an abbreviated version of the month name instead. If the optional argument is omitted, the value of an internal configuration option is used, if present - contact your Genesys representative for details. Otherwise, the platform locale is used.

MonthNameShort String

Argument	Description
First argument	If it is a numeric value (1 to 12), it is converted to the appropriate month name. If it is a Date/Time, the month number is extracted and converted.
Second argument	Optional. The locale that must be used to format the first argument. Some examples include: en_US for English (United States), en_GB for English (United Kingdom), and fr_FR for French (France). The value must be in the following format: _

Second

Second (*DateTime)*

Returns the numeric second component of a `Date/Time (0–59).`

Time

Time ([*Hour,* [*Minute,* [*Second*]]])

Returns a `Date/Time` constructed from individual time components. The date components of the result (year, month, and day) are set to the current system date. The optional arguments default to `0` if omitted. If all the optional arguments are omitted, then the time is set to the current system time.

> **Important**
> The examples in the Examples of Time String table assume that the current system date is November 23, 2003, @ 09:03:10.

Examples of Time String

Example	Result
	2003-11-23 09:03:10
	2003-11-23 15:00:00
	2003-11-23 15:23:10

TimeGMT

TimeGMT()

Returns a `Date/Time` set to the current system time and converted to GMT (Greenwich mean time), also called Universal Time Coordinated, or UTC.

ToTimeZoneDate

ToTimeZoneDate*(DateString, TimeZoneString)*

Returns a `Date/Time` constructed from a string and a time zone. This date is constructed by parsing the string and using the specified time zone . Examples include the following:

-
-

Weekday

Weekday (*DateTime*)

Returns the numeric weekday component of a `Date/Time` (`0` = Sunday to `6` = Saturday).

WeekdayName

WeekdayName*(Arg[, String])*

Converts a number of a Date/Time to a weekday name. If the optional argument is omitted, the value of an internal configuration option is used, if present - contact your Genesys representative for details. Otherwise, the platform locale is used.

WeekdayName String

Argument	Description
First argument	If it is a numeric value (0 to 6), it is converted to the appropriate weekday name. If it is a Date/Time, the weekday number is extracted and converted.
Second argument	The locale that must be used to format the first argument. Some examples include: en_US for English (United States), en_GB for English (United Kingdom), andfr_FR for French (France). The value must be in the following format: _

WeekdayNameShort

WeekdayNameShort*(Arg[, String])*

The same as WeekdayName but this returns an abbreviated weekday name instead. If the optional

argument is omitted, the value of an internal configuration option is used, if present - contact your Genesys representative for details. Otherwise, the platform locale is used.

WeekdayNameShort String

Argument	Description
First argument	If it is a numeric value (0 to 6), it is converted to the appropriate weekday name. If it is a Date/Time, the weekday number is extracted and converted.
Second argument	The locale that must be used to format the first argument. Some examples include: en_US for English (United States), en_GB for English (United Kingdom), and fr_FR for French (France). The value must be in the following format: _

Year

Year (*DateTime*)

Returns the numeric year component of a `Date/Time` with the century.

YearShort

YearShort (*DateTime*)

Returns the numeric year component of a `Date/Time` without the century (`0`– `99`).

Type conversion

Contents

- 
- Administrator

Learn about the type conversions you can use in field codes.

Related documentation:

-

Field codes can use the type conversions detailed on this page.

Bool

Bool(*Arg*, [*Default*])

Returns a Boolean converted from a number or a string.

Arguments

Name	Description
Arg	If a number, then converts 0 to False and nonzero to True. If a string, then converts Off, No, and False to False, and On, Yes, and True to True. If another string, then returns Default. If Default is omitted, then returns False.

Examples

Example	Result
	False
	True
	True
	False
	False
	True

Num

Num (*String*,*[String]*)

Returns a number converted from a string.

Arguments

Name	Description
First argument	The string to be converted. May be expressed in scientific notation. Returns 0 if the string is not recognizable as a number. Ignores nonnumeric characters following the number.
Second argument	Optional. The locale that must be used to parse the first argument. Some examples include: en_US for English (United States), en_GB for English (United Kingdom), and fr_FR for French (France). If omitted, the value of an internal configuration option is used, if present - contact your Genesys representative for details. Otherwise, the platform locale is used.

Examples

For clarity, the results shown in the table appear with three digits after the decimal point and always in the en_US format. Default number formatting shows no digits after the decimal point. Use the Text function (see Text) or format operator (%) to override the default formatting.

Example	Result
	10.000
	10.000 (Assuming the locale is en_US.)
	10.000 (Note the comma-decimal separator in the first argument.)
	0.120
	1220.000 (Assuming the locale is en_US.)
	1220.000 (Note the comma-decimal separator in the first argument.)
	0.000

Text

Text (*Arg[,Pattern[,String]]*) or **Text (*Arg:Pattern*)**

Returns a string converted from an argument of any data type. Use the format operator (:) as shorthand for this function.

Arguments

Name	Description
Arg	The value to be converted
Pattern	Optional. The picture string to use for formatting. If omitted, default formatting is used. The syntax of the picture string depends on the data type. See Number Formatting (Arg is a Number).
String	Optional. The locale that must be used to parse the first argument. Some examples include: en_US for English (United States), en_GB for English (United Kingdom), and fr_FR for French (France). If omitted, the value of an internal configuration option is used, if present - contact your Genesys representative for details. Otherwise, the platform locale is used.

Number Formatting (Arg is a Number)

If Arg is a number, then the regular expression syntax of the optional pattern string is as follows:

`#*.?#*`

Where:

The pound sign (#) represents a digit. Any number of #s, including 0 may appear before the decimal character. Specify the minimum number of digits that should appear to the left of the decimal. If the integer part of the formatted number contains fewer than the specified number of digits, the number is padded with leading zeros.

Any number of #s, including 0, may appear after the decimal character. Specify the precision of the fractional part of the number. The number is rounded to the specified precision.

Only the decimal separator in the result is locale dependent (there is no grouping separator).

The Examples of Number Formatting table contains some examples.

Examples of Number Formatting

Pattern	Arg Value	Locale	Result
""	0	en_US	"0"
""	123.456	en_US	"123"
"#"	0	en_US	"0"
"##"	0	en_US	"00"
"##"	123.456	en_US	"123"
"#."	0	en_US	"0. "
"#."	123.456	en_US	"123. "
".##"	0	en_US	".00"
".##"	0.456	en_US	".46"
".##"	123.456	en_US	"123.46"
".##"	20000.456	en_US	"20000.46" (Note the decimal point separator in the result.)
".##"	123.456	fr_FR	"123,46" (Note the comma-decimal separator in the result.)
".##"	20000.456	fr_FR	"20000,46" (Note the comma-decimal separator in the result.)

Duration Formatting (Arg is a Number)

If Arg is a number, then the regular expression syntax of the optional pattern string is as follows:

`()?.?#*`

Where:

represents a duration and can be any of the sequences in the following list. Upper- or lowercase letters are accepted.

- HH
- HH:MM
- HH:MM:SS
- MM
- MM:SS
- SS
- H
- H:MM
- H:MM:SS
- M
- M:SS
- S

may be followed by a .## string, which specifies the precision of the last element of the duration. Any C or % suffixes are ignored. When you format a value as a duration, the value is always assumed to be expressed in days.

The pound sign (#) represents a digit. Any number of #s, including 0, may appear before the decimal character and specify the minimum number of digits that should appear to the left of the decimal. If the integer part of the formatted number contains fewer than the specified number of digits, the number is padded with leading zeroes.

Any number of #s, including 0, may appear after the decimal character and specify the precision of the fractional part of the number. The number is rounded to the specified precision.

The Examples of Duration Formatting table contains some examples.

Examples of Duration Formatting

Pattern	Arg Value	Locale	Result
"HH"	10.5083	en_US	"11"
"HH.## "	10.5083	en_US	"10.51"
"HH:MM"	10.5083	en_US	"10:30"
"HH:MM.# "	10.5083	en_US	"10:30.5"

Pattern	Arg Value	Locale	Result
"HH:MM:SS"	10.5083	en_US	"10:30:30"
"MM"	10.5083	en_US	"630"
"MM.## "	10.5083	en_US	"630.50"
"MM:SS"	10.5083	en_US	"630:30"
"SS"	10.5083	en_US	"37830"

Currency Formatting (Arg is a Number)

If Arg is a number, then the regular expression syntax of the optional parameter string is as follows:

`#*.?#*[Cc]`

Where:

A C or a c means format as currency. The grouping separator, the decimal separator, and the currency sign in the result are locale dependent.

The Examples of Currency Formatting table contains some examples.

Examples of Currency Formatting

Pattern	Arg Value	Locale	Result
"C"	12.34	en_US	"$12.34"
"C"	-12.34	en_US	"($12.34)"
"#.#C"	12.34	en_US	"$12.3"
"#.#C"	-12.34	en_US	"($12.3)"
"C"	12.34	en_GB	"£12.34 "
"C"	-12.34	en_GB	"-£12.34 "

Pattern	Arg Value	Locale	Result
"#.#C"	12.34	en_GB	"£12.3 "
"#.#C"	-12.34	en_GB	"-£12.3 "
".##C"	20000.456	en_US	"$20,000.46" (Note the comma grouping separator and point decimal separator in the result.)
".##C"	20000.456	fr_FR	"20 000,46 €" (Note the decimal comma separator in the result.)

Percentage Formatting (Arg is a Number)

If Arg is a number, then the regular expression syntax of the optional pattern string is as follows:

`#*.?#*%`

Where:

The percent sign (%) means multiply by 100 and append the locale-dependent sign for percent values. If the % appears by itself, the formatter rounds to the nearest integral value and omits a decimal point (equivalent to the format #%).

The grouping separator, the decimal separator, and the percent sign in the result are locale dependent.

The Examples of Percentage Formatting table contains some examples.

Examples of Percentage Formatting

Pattern	Arg Value	Locale	Result
"%"	0	en_US	"0%"
"%"	0.123456	en_US	"12%"
"#.##%"	0.123456	en_US	"12.35%"
"#.##%"	0.123456	fr_FR	"12,35%" (Note the comma-decimal

Pattern	Arg Value	Locale	Result
			separator in the result.)

Date/Time Formatting

Use elements shown in the Date/Time Pattern Letters table to construct a Date/Time pattern string. The letters must be in uppercase or lowercase, as shown in the table (for example, MM not mm). Characters that are not picture elements, or that are enclosed in single quotation marks, will appear in the same location and unchanged in the output string.

Date/Time Pattern Letters

Element	Meaning
d	Day of month as digits, with no leading zero for single-digit days
dd	Day of month as digits, with leading zero for single-digit days
ddd	Day of week as a three-letter abbreviation
dddd	Day of week as its full name
M	Month as digits, with no leading zero for single-digit months
MM	Month as digits, with leading zero for single-digit months
MMM	Month as a three-letter abbreviation
MMMM	Month as its full name
y	Year as last two digits, but with no leading zero for years less than 10
yy	Year as last two digits, but with leading zero for years less than 10
yyyy	Year represented by full four digits

Element	Meaning
h	Hours, with no leading zero for single-digit hours; 12-hour clock
hh	Hours, with leading zero for single-digit hours; 12-hour clock
H	Hours, with no leading zero for single-digit hours; 24-hour clock
HH	Hours, with leading zero for single-digit hours; 24-hour clock
m	Minutes, with no leading zero for single-digit minutes
mm	Minutes, with leading zero for single-digit minutes
s	Seconds, with no leading zero for single-digit seconds
ss	Seconds, with leading zero for single-digit seconds
tt	Time-marker string, such as AM or PM

The examples in the Examples of Date/Time Formatting table assume that the date being formatted is August 6, 2003, @ 15:05:10:

Examples of Date/Time Formatting

Pattern	Locale	Result
"MMMM d, yyyy @ hh:mm:ss tt"	en_US	"August 6, 2003 @ 03:05:10 PM"
"MMMM dd, yyyy @ HH:mm:ss"	en_US	"August 06, 2003 @ 15:05:10"
"dd MMMM yyyy HH:mm:ss"	fr_FR	"06 aout 2003 15:05:10"
"MMM d, yy @ h:mm:ss tt"	en_US	"Aug 6, 03 @ 3:05:10 PM"
"M/dd/yy"	en_US	"8/06/03"

Boolean Formatting

A Boolean picture string is simply two words separated by a comma. The first word is used if the Boolean value is True, and the second is used otherwise.

The Examples of Boolean Formatting table shows some examples:

Examples of Boolean Formatting

Field Code	Result

	"Yup"
	"No"
	"peach"

String Formatting

Picture strings do not apply to string values. Strings are always output unchanged. If you want to output a piece of a string, or change the case, then you can use one of the string-manipulation functions previously described.

Mathematical functions

Contents

- 
- Administrator

Learn about the mathematical functions you can use in field codes.

Related documentation:

-

Field codes can use the mathematical functions detailed on this page.

Abs

Abs (*Number*)

Returns the absolute value of a number. The absolute value of a number is the number without regard to its sign.

Ceil

Ceil (*Number*)

Returns the ceiling of a number. The ceiling of a number is the smallest integer that is greater than or equal to that number.

Floor

Floor (*Number*)

Returns the floor of a number. The floor of a number is the largest integer that is less than or equal to that number.

Miscellaneous functions

Contents

- 
- Administrator

Learn about the miscellaneous functions you can use in field codes.

Related documentation:

-

Field codes can use the miscellaneous functions detailed on this page.

If

If (*Boolean, TrueResult, FalseResult*)

Returns either the second or the third argument, depending on the value of the first (Boolean) argument.

IsBoolean

IsBoolean (*Arg*)

Returns True if the data type of the argument is Boolean; otherwise, it returns False.

IsDateTime

IsDateTime (*Arg*)

Returns True if the data type of the argument is Date/Time, and False otherwise.

IsNumber

IsNumber (*Arg*)

Returns True if the data type of the argument is number, and False otherwise.

IsString

IsString (*Arg*)

Returns True if the data type of the argument is string, and False otherwise.

Type

Type (*Arg*)

Returns the type name (String, Boolean, and so on) of its argument.

Agent object

- 
- Administrator

Learn about the agent object you can access using system variables in field codes.

Related documentation:

-

The Agent object is associated with the Interaction object. For an automated reply, this object is the agent whose login name equals the E-mail Server autobot-agent-login option. Contact your Genesys representative to configure this feature.

Name	Description	Syntax
FirstName	Returns this agent's first name.	Agent.FirstName
LastName	Returns this agent's last name.	Agent.LastName
FullName	Returns this agent's full name (first and last).	Agent.FullName
Signature	Returns this agent's signature.	Agent.Signature

Contact object

- 
- Administrator

Learn about the contact object you can access using system variables in field codes.

Related documentation:

-

The Contact object is associated with the current EmailIn interaction. The properties include:

Name	Description	Syntax
Id	Returns this contact’s ID.	Contact.Id
FirstName	Returns this contact’s first name.	Contact.FirstName
LastName	Returns this contact’s last name.	Contact.LastName
FullName	Returns this contact’s full name (first and last).	Contact.FullName
Title	This contact’s title (for example, Mr., Ms., and so on).	Contact.Title
PrimaryEmailAddress	Returns this contact’s primary e-mail address.	Contact.PrimaryEmailAddress
PrimaryPhoneNumber	Returns this contact’s primary phone number.	Contact.PrimaryPhoneNumber

Interaction object

- 
- Administrator

Learn about the interaction object you can access using system variables in field codes.

Related documentation:

-

The Interaction object is the currently processed interaction that is built from a standard response and includes field codes.

- For Acknowledgement, Redirect, Autoresponse, Chat Transcript, Forward, and Reply From External Resource strategy objects, this Interaction object handles EmailIn.
- For the Send object, which only supports field codes for this Subject, this Interaction object handles EmailOut.
- This distinction affects the the FromAddress and ToAddresses properties.

The properties for this object include:

Name	Description	Syntax
Id	Returns the interaction's ID.	Interaction.Id
DateCreated	Returns the Date/Time at which this Interaction was created in the system.	Interaction.DateCreated
Subject	Returns the Subject of this Interaction.	Interaction.Subject
ToAddress	Returns the recipient (To field) of this Interaction.	Interaction.ToAddress
FromAddress	Returns the originator (From field) of this Interaction.	Interaction.FromAddress
AttachedData	Returns the attached data (Interaction Attribute) value associated with a specified key. The value can be either a string or a number. For example:	Interaction.AttachedData (“Key”)

Name	Description	Syntax
	• Interaction.AttachedData (“ParentId”) • Interaction.AttachedData (“Language”)	
TimeZone	Returns the time zone of the parent interaction (Interaction in general). The value is a string formatted as“GMT”, “GMT+”hh.mm, or “GMT-”hh.mm. For example: • GMT+01.00 indicates a Paris time zone. • GMT-04.00 indicates a Canada east coast (Maritimes) time zone. • GMT-05.00 indicates an eastern U.S./Canada time zone.	Interaction.TimeZone