eServices Field Codes Reference Manual

Genesys Engage Digital (eServices) 8.1.4

1/11/2022

Table of Contents

Genesys eServices Field Codes Reference Manual

This guide describes eServices Field Codes that are used in standard responses.

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 <$Contact.FirstName$>,

…...

<$Agent.Signature$>
```

This response has two field codes. When an agent inserts this response into an e-mail, the first field code, `<$Contact.FirstName$>`, is replaced by the contact's first name as it appears in Universal Contact Server. The second field code, `<$Agent.Signature$>`, 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 e-mail 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
```

See the "Field Codes" section of the Knowledge Manager User's Guide for information on how to use field codes in standard responses.

The remainder of this manual provides the following reference sections:

- Escape Codes and Sequences
- Data Types
- Operator Precedence

Escape Codes and Sequences

Since the delimiters <$ and $> have special meanings when they appear in field codes, you cannot include them “as is” in a standard response. If you want to write a standard response that includes either or both of these field code delimiters, you must insert a space between the two symbols that make up each delimiter. For example, here is a valid standard response:

```
These field codes are great! You begin them with < $ and end
them with $ >.
```

Data Types

The eServices Field Codes include the following data types:

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
< LT	Less Than	2 < 3	True

Operator	Description	Example	Result
<= LE	Less Than or Equal To	2 <= 3	True
= == EQ	Equal To	2 = 3	False
<> != NE	Not Equal To	2 <> 3	True
:	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

Escape	Translates to
\"	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
<LT	Less Than	"A" < "B"	True
<=LE	Less Than or Equal To	"A" <= "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, <, =, and so on).
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`

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

Operator Precedence

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

<toggledisplay linkstyle font-size:larger showtext="[+] Operator Precedence Table" hidetext="[-] Operator Precedence Table">

Operator Precedence Table

Operator
+ [Unary], - [Unary]
^
• , /, Mod
+, -
< (LT), <= (LE), > (GT), >= (GE), = (==, EQ), <> (!=, NE)
Not (!) [Unary]
And (&&)
XOr

Operator
Or (||)
:

Named Constants

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 Equivalents Table

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

Functions

The eServices Field Codes include the following functions:

String Functions

The eServices Field Codes include the following string functions:

[+] Find

Find

Description
Finds a substring within a string. Returns the `0`- based character position of the found substring. Returns –1 if the substring is not found.
Syntax
Find(*SearchIn, SearchFor*)

Find String

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

Remarks

Examples of Find String

Example	Result
<$Find("Hello, World!", "H")$>	`0`
<$Find("Hello, World!", "lo")$>	`3`
<$Find("Hello, World!", "Qbert")$>	`-1`

[+] Left

Left

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

Syntax
Left(*String, Number*)

Left String

Argument	Description
String	The string from which the leftmost 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.

Remarks

Examples of Left String

Example	Result
<$Left("Hello, World!", 5)$>	"Hello"
<$Left("Hello, World!", 0)$>	""
<$Left("Hello, World!", 25)$>	"Hello, World!"

[+] Length

Length

Description
Returns the length of a string.
Syntax
Length(*String*)
Remarks

Example of Length String

Example	Result
<$Length("Hello")$>	5

[+] Mid

Mid

Description
Returns a specified substring of a string.
Syntax
Mid(*String, Start, Length*)

Mid 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.

Remarks

Examples of Mid String

Example	Result
<$Mid("Hello, World!", 2, 3)$>	`"llo"`
<$Mid("Hello, World!", 25, 5)$>	`""`
<$Mid("Hello, World!", 7, 25)$>	`"World!"`

[+] Replace

Replace

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

Syntax
Replace(*String, Find, ReplaceWith*)

Replace String

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

Remarks

Examples of Replace String

Example	Result
<$Replace("Hello", "l", "*")$>	"He**o"
<$Mid("Hello", "j", "*")$>	"Hello"
<$Mid("Hello", "Hello", "")$>	""

[+] Right

Right

Description
Returns a string containing a specified number of characters from the right side of a specified string.
Syntax
Right(*String, Number*)

Right String

Argument	Description
String	The string from which the rightmost 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.

Remarks

Examples of Right String

Example	Result
<$Right("Hello, World!", 5)$>	`"orld!"`
<$Right("Hello, World!", 0)$>	`""`
<$Right("Hello, World!", 25)$>	`"Hello, World!"`

[+] To Lower

ToLower

Description
Returns a string that has been converted to lowercase.
Syntax
ToLower(*String*)
Remarks

Example of ToLower String

Example	Result
<$ToLower("Hello, World!")$>	`"hello, world!"`

[+] ToUpper

ToUpper

Description
Returns a string that has been converted to uppercase.
Syntax
ToUpper(*String*)

Remarks

Example of ToUpper String

Example	Result
<$ToUpper("Hello, World!")$>	"HELLO, WORLD!"

[+] Trim

Trim

Description
Returns a copy of a specified string without specified leading or trailing characters.
Syntax
Trim(*String,* [*CharSet*])

Trim String

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.

Remarks

Examples of Trim String

Example	Result
<$Trim(" Howdie ")$>	"Howdie"
<$Trim("Howdie", "Howd")$>	"ie"
<$Trim("Howdy", "y")$>	"Howd"

[+] TrimLeft

TrimLeft

Description
The same as `Trim`, except it trims only leading characters.
Syntax
TrimLeft(*String,* [*CharSet*])

[+] TrimRight

TrimRight

Description
The same as `Trim`, except it trims only trailing characters.
Syntax
TrimRight(*String,* [*CharSet*])

[+] FullName

Wrap

Description
Returns a string that has been word-wrapped to a specified line length.
Syntax
Trim(*String, LineLength,* [*LinePrefix,* [*Eol*]])

Wrap String

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.

Remarks
Example:

```
<$Wrap(> "Once upon a midnight dreary",<
11,<
">",<
"*\r\n")$>
```

Result:

```
>Once upon*<
>a midnight*<
>dreary*
```

Date and Time Functions

The eServices Field Codes include the following date/time functions:

[+] Date

Date

Description
Returns a `Date/Time` constructed from individual components or a string.
Syntax
Date(*Year, Month, Day* [, *Hour[, Minute[, Second*]]])
Or
Date(String[, String])

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). See See Values for fieldcode-format-locale Option for a complete list.

> **Important**
> Date(String[, String]) is not recommended. See the “Remarks” section.

Remarks

- When using the first syntax function, the optional arguments each default to `0` if omitted. For example, `<$Date(1965, 11, 23)$>` is equivalent to `<$Date(1965, 11, 23, 0, 0, 0)$>`.
- When using the second syntax function, the date is constructed by parsing the first string. If the optional argument is omitted, first the E-mail Server `fieldcode-format-locale` option (See fieldcode-format-locale) in the `email-processing` section is used if present. Otherwise, the platform locale is used. For example:
  - `<$Date("November 23, 1965 9:03 AM")$>` if the `fieldcode-format-locale` option or platform locale is set to `en_US`.
  - `<$Date(“23 novembre 1965 21:03:00”, “fr_FR”)$>`

> **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, `<$Date("November 23, 1965, at 9:03 AM")$>` 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.

[+] Day

Day

Description
Returns the numeric day component of a `Date/Time (1 to 31)`.
Syntax
Day(*DateTime*)

[+] Hour12

Hour12

Description
Returns the numeric hour component of a `Date/Time` based on a 12-hour clock `(1 to 12)`.
Syntax
Hour12(*DateTime*)

[+] Hour24

Hour24

Description
Returns the numeric hour component of a `Date/Time` based on a 24-hour clock `(0 to 23)`.
Syntax
Hour24(*DateTime*)

[+] IsAm

IsAm

Description

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

[+] IsPm

IsPm

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

[+] Minute

Minute

Description
Returns the numeric minute component of a `Date/Time (0–59)` .
Syntax
Minute(*DateTime*)

[+] Month

Month

Description
Returns the numeric month component of a `Date/Time (1–12)` .
Syntax
Month(*DateTime*)

[+] MonthName

MonthName

Description
Converts a month number or a Date/Time to a month name.
Syntax
MonthName*(Arg[, String])*

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). See See Values for fieldcode-format-locale Option for a complete list.

Remarks
If the optional argument is omitted, first the E-mail Server `fieldcode-format-locale` option (See fieldcode-format-locale) in the `email-processing` section is used if present. Otherwise, the platform locale is used.

[+] MonthNameShort

MonthNameShort

Description
The same as the MonthName, but this returns an abbreviated version of the month name instead.

Syntax
MonthNameShort*(Arg[, String])*

MonthNameShort String

Argument	Description
First argument	If it is a numeric value (1 to 12), it is converted to the appropriate abbreviated 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). See See Values for fieldcode-format-locale Option for a complete list.

Remarks
If the optional argument is omitted, first the E-mail Server `fieldcode-format-locale` option (See fieldcode-format-locale) in the `email-processing` section is used if present. Otherwise, the platform locale is used.

[+] Second

Second

Description
Returns the numeric second component of a `Date/Time` `(0–59)`.
Syntax
Second (*DateTime*)

[+] Time

Time

Description
Returns a `Date/Time` constructed from individual time components.
Syntax
Time ([*Hour,* [*Minute,* [*Second*]]])
Remarks
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
<$Time()$>	`2003-11-23 09:03:10`
<$Time(15)$>	`2003-11-23 15:00:00`
<$Time(15, 23, 10)$>	`2003-11-23 15:23:10`

[+] TimeGMT

TimeGMT()

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

[+] ToTimeZoneDate

ToTimeZoneDate

Returns a Date/Time constructed from a string and a time zone.
Syntax
ToTimeZoneDate*(DateString, TimeZoneString)*
Remarks
This date is constructed by parsing the <DateString> string and using the specified time zone <TimeZoneString> . Examples include the following:
<$ToTimeZoneDate(Date(“November 23, 1965 9:03 AM”), “America/Los_Angeles”)$>
<$ToTimeZoneDate(Date(“11/23/65 9:03:00”), “Europe/Paris”)$>

[+] Weekday

Weekday

Description
Returns the numeric weekday component of a Date/Time (0 = Sunday to 6 = Saturday).
Syntax
Weekday (*DateTime*)

[+] WeekdayName

WeekdayName

Description
Converts a number of a Date/Time to a weekday name.
Syntax
WeekdayName*(Arg[, String])*

WeekdayName String

|- valign="top" | rowspan="1" colspan="1" | Argument
| rowspan="1" colspan="1" | Description
|- valign="top" | rowspan="1" colspan="1" | First argument
| rowspan="1" colspan="1" | 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.
|- valign="top" | rowspan="1" colspan="1" | Second argument

| rowspan="1" colspan="1" | 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), andfr_FR for French (France). See See Values for fieldcode-format-locale Option for a complete list.
|} **Remarks**
If the optional argument is omitted, first the E-mail Server `fieldcode-format-locale` option (See fieldcode-format-locale) in the `email-processing` section is used if present. Otherwise, the platform locale is used.

[+] WeekdayNameShort

WeekdayNameShort

Description
The same as WeekdayName but this returns an abbreviated weekday name instead.
Syntax
WeekdayNameShort*(Arg[, String])*

WeekdayNameShort String

|- valign="top" | rowspan="1" colspan="1" | Argument
| rowspan="1" colspan="1" | Description
|- valign="top" | rowspan="1" colspan="1" | First argument
| rowspan="1" colspan="1" | If it is a numeric value (`0` to `6`), it is converted to the appropriate abbreviated weekday name. If it is a Date/Time, the weekday number is extracted and converted.
|- valign="top" | rowspan="1" colspan="1" | Second argument
| rowspan="1" colspan="1" | 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). See See Values for fieldcode-format-locale Option for a complete list.
|} **Remarks**
If the optional argument is omitted, first the E-mail Server `fieldcode-format-locale` option (See fieldcode-format-locale) in the `email-processing` section is used if present. Otherwise, the platform locale is used.

[+] Year

Year

Description
Returns the numeric year component of a `Date/Time` with the century.
Syntax
Year (*DateTime*)

[+] YearShort

YearShort

Description

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

Type Conversion

The eServices Field Codes use the following type conversions:

Bool

[+] Bool

Description
Returns a Boolean converted from a number or a string.
Syntax
Bool(*Arg,* [*Default*])

Bool String

Argument	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`.

Remarks

Examples of Bool String

Example	Result
<$Bool(0)$>	`False`
<$Bool(25.23)$>	`True`
<$Bool("Yes")$>	`True`
<$Bool("off", True)$>	`False`
<$Bool("Asteroids")$>	`False`
<$Bool("Asteroids", True)$>	`True`

Num

[+] Num

Description
Returns a number converted from a string.
Syntax
Num (*String[, String]*)

Num String

Argument	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). See the Values for `fieldcode-format-locale` Option in the *eServices 8.1 Reference Manual* for a complete list.

Remarks

If the optional argument is omitted, first the E-mail Server `fieldcode-format-locale` option (see option description in the *eServices 8.1 Reference Manual*) in the `email-processing` section is used if present. Otherwise, the platform locale is used.
For clarity, the results shown in the Examples of Num String 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 Field Codes in Standard Responses Reference: Text) or format operator (`%`) to override the default formatting.

Examples of Num String

Example	Result
<$Num("10")$>	`10.000`
<$Num("10.00")$>	`10.000` (Assuming the locale is `en_US.`)
<$Num("10,00", "fr_FR")$>	`10.000` (Note the comma-decimal separator in the first argument.)
<$Num("12e-2")$>	`0.120`
<$Num("12.2e2Zork")$>	`1220.000` (Assuming the locale is en_US .)

Example	Result
<$Num("12,2e2Zork", "fr_FR")$>	1220.000 (Note the comma-decimal separator in the first argument.)
<$Num("Zaxxon")$>	0.000

Text

[+] Text

Text

Description
Returns a string converted from an argument of any data type. Use the format operator (:) as shorthand for this function.
Syntax
Text (*Arg[, Pattern[, String]]*)
or
Arg:Pattern

Text String

Argument	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 - “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). See Values for the `fieldcode-format-locale` Option in the *eServices 8.1 Reference Manual* for a complete list. If the optional argument is omitted, first the E-mail Server `fieldcode-format-locale` option in the email-processing section is used, if present. Otherwise, the platform locale is used.

[+] Number Formatting

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"

Pattern	Arg Value	Locale	Result
".##"	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

Duration Formatting (Arg is a Number)

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

```
(<dur>).?#*
```

Where:
<dur> 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

> <dur> 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"
"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

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 "
"#.#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

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 separator in the result.)

[+] Date/Time Formatting

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

Element	Meaning
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
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"

Pattern	Locale	Result
"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

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
<$Text(True, "Yup,Nope")$>	"Yup"
<$Text(False, "Si,No")$>	"No"
<$Text(False, "walnut,peach")$>	"peach"

[+] String Formatting

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

The eServices Field Codes include the following mathematical functions:

Abs

[+] Abs

Description
Returns the absolute value of a number.
Syntax
Abs (*Number*)
Remarks
The *absolute* value of a number is the number without regard to its sign.

Ceil

[+] Ceil

Description
Returns the ceiling of a number.
Syntax
Ceil (*Number*)
Remarks
The *ceiling* of a number is the smallest integer that is greater than or equal to that number.

Floor

[+] Floor

Description
Returns the floor of a number.
Syntax
Floor (*Number*)
Remarks
The *floor* of a number is the largest integer that is less than or equal to that number.

Miscellaneous Functions

The eServices Field Codes include the following miscellaneous functions:

If

[+] If

Description
Returns either the second or the third argument, depending on the value of the first (Boolean) argument.
Syntax
If (*Boolean, TrueResult, FalseResult*)

IsBoolean

[+] IsBoolean

Description
Returns `True` if the data type of the argument is `Boolean`; otherwise, it returns `False`.
Syntax
IsBoolean (*Arg*)

IsDateTime

[+] IsDateTime

Description
Returns `True` if the data type of the argument is `Date/Time`, and `False` otherwise.
Syntax
IsDateTime (*Arg*)

IsNumber

[+] IsNumber

Description
Returns `True` if the data type of the argument is `number`, and `False` otherwise.
Syntax
IsNumber (*Arg*)

IsString

[+] IsString

Description
Returns `True` if the data type of the argument is `string`, and `False` otherwise.
Syntax
IsString (*Arg*)

Type

[+] Type

Description
Returns the type name (`String, Boolean`, and so on) of its argument.
Syntax
Type (*Arg*)

Objects

The following objects can be accessed through eServices Field Codes:

- Agent Object
- Contact Object
- Interaction Object

Agent Object

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 (see option description in the *eServices 8.1 Reference Manual*) in its `email-processing` section.

FirstName

[+] FirstName

Description

Returns this agent's first name.

Syntax

Agent.FirstName

LastName

[+] LastName

Description

Returns this agent's last name.

Syntax

Agent.LastName

FullName

[+] FullName

Description

Returns this agent's full name (first and last).

Syntax

Agent.FullName

Signature

[+] Signature

Description

Returns this agent’s signature.

Syntax

Agent.Signature

Contact Object

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

Id

[+] Id

Description

Returns this contact’s ID.

Syntax

Contact.Id

FirstName

[+] FirstName

Description

Returns this contact’s first name.

Syntax

Contact.FirstName

LastName

[+] Lastname

Description

Returns this contact’s last name.

Syntax

Contact.LastName

FullName

[+] FullName

Description

Returns this contact’s full name (first and last).

Syntax

Contact.FullName

Title

[+] Title

Description

This contact’s title (for example, Mr., Ms., and so on).

Syntax

Contact.Title

PrimaryEmailAddress

[+] PrimaryEmailAddress

Description

Returns this contact’s primary e-mail address.

Syntax

Contact.PrimaryEmailAddress

PrimaryPhoneNumber

[+] PrimaryPhoneNumber

Description

Returns this contact’s primary phone number.

Syntax

Contact.PrimaryPhoneNumber

Interaction Object

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:

Id

[+] Id

Description

Returns the `Interaction's` ID.

Syntax

Interaction.Id

DateCreated

[+] DateCreated

Description

Returns the `Date/Time` at which this `Interaction` was created in the system.

Syntax

Interaction.DateCreated

Subject

[+] Subject

Description

Returns the `Subject` of this `Interaction`.

Syntax

Interaction.Subject

ToAddress

[+] ToAddress

Description

Returns the recipient (`To` field) of this `Interaction`.

Syntax

Interaction.ToAddress

> **Important**
> For the Send strategy object, this syntax translates into the current `EmailOut.ToAddresses`. For the `Acknowledgement`, `Redirect`, `Autoresponse`, `Chat Transcript`, `Forward`, and `Reply From External Resource` strategy objects, this translates into the current `EmailIn.ToAddresses`.

FromAddress

[+] FromAddress

Description

Returns the originator (`From` field) of this `Interaction`.

Syntax

Interaction.FromAddress

> **Important**
> For the Send strategy object, this syntax translates into the current `EmailOut.ToAddresses`. For the `Acknowledgement, Redirect, Autoresponse, Chat Transcript, Forward`, and `Reply From External Resource` strategy objects, this translates into the current `EmailIn.FromAddresses`.

AttachedData

[+] AttachedData

Description

Returns the attached data (`Interaction Attribute`) value associated with a specified key. The value can be either a string or a number.

Syntax

Interaction.AttachedData (“Key”)

Example

```
Interaction.AttachedData (“ParentId”)
Interaction.AttachedData (“Language”)
```

TimeZone

[+] TimeZone

Description

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`.

Syntax

Interaction.TimeZone.

Examples

`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.