eServices Manager Plug-in for GAX

Functions, Arguments, and Operators

12/19/2025

Functions, Arguments, and Operators

Functions

Screening rules can use three basic functions:

- `Find("<text>")`, where `<text>` is a text string. It returns the result `true` if the interaction contains the exact string between quotes, ignoring case.
- `RegExFind("<regular expression>")`, where `<regular expression>` is a regular expression (see Regular Expressions). It returns the result `true` if the interaction contains any string that matches the regular expression between quotes.
- `RegExMatch("<regExp>")`, where `<regular expression>` is a regular expression. It returns the result `true` only if the entire content of the interaction matches the regular expression between quotes.

> **Important**
>
> RegExFind and RegExMatch are the same except that RegExFind looks for a match anywhere in the body of the interaction, whereas RegExMatch demands that the entire body of the interaction match the regular expression.

Arguments

All functions have one required argument, which must appear between double quotation marks, as represented above (`<text>`) or (`<regular expression>`). This required argument can be followed by one or two optional arguments, depending on the function. The full form of each function, including all arguments, is as follows:

- `Find("<text>", <IgnoreCase>)`
- `RegExFind("<regular expression>",<"key">,<IgnoreCase>)`
- `RegExMatch("<regular expression>",<IgnoreCase)`

IgnoreCase

The IgnoreCase argument must be a Boolean value (*true* or *false*). All three functions ignore case in

searches unless you include the IgnoreCase argument with a value of `false`.

For example:

- `Find("pacific")` finds *Pacific* and *pacific.*
- `Find("Pacific",false)` finds *Pacific* but not *pacific.*

You can also substitute `true` for `false`—for example, `Find("Pacific",true)`—which means that case is ignored. So `Find("Pacific",true)` is the same as `Find("Pacific")`.

Key

The key argument must be a string. If this argument is present, the system creates a key-value pair with the following characteristics:

- The key name is the string specified by the key argument, prefixed by `ScrKey_`.
- The value is the material that the screening rule matches.

The system then adds this key-value pair to the interaction's attached data. For example, `RegExFind("[A-Z]\d\d\d","ID_code",false)`:

1. Finds strings consisting of a capital letter followed by three digits (see Regular Expressions).
2. Attaches to the interaction a key-value pair called `ScrKey_ID_code` whose value is A123, X005, M999, or whatever the function found in this interaction to match the regular expression.

Operators

Operators are of two types:

- Binary operators join two functions.
- Unary operators operate on a single function.

The operators are as follows:

&& is the binary operator "and". For example,

```
Find("interest rate") && Find("APR",false)
```

matches a message only if it includes both "interest rate" and "APR."

|| is the binary operator "or." For example,

```
Find("station wagon") || Find("convertible")
```

matches any message that includes either "station wagon" or "convertible" (or "Station Wagon" or "station Wagon" or "Convertible").

! is the unary operator "not." For example,

```
!Find("windows")
```

matches any message that does not include the word "windows."

You can combine `!` with a binary operator. For example,

```
Find("bird") && !Find("goose")
```

matches any message that includes "bird" but does not include "goose."

Operator Precedence

`p && q || r` is parsed as `(p && q) || r`. For example, consider:

```
Find("debt") && Find("income") || Find("profit")
```

To paraphrase, this screening rule is basically "find X or find Y," where X is "debt" and "income," and Y is "profit." It matches both "debt exceeds income" and "profits are fantastic".

You can modify the default precedence by the explicit use of parentheses; for example:

```
Find("debt") && (Find("income") || Find("profit"))
```

This screening rule is basically "find X and find Y," where X is "debt" and Y is either "income" or "profit." It matches both "debt exceeds income" and "debts impact profit."