Designer User's Guide

Manage field codes

1/29/2022

Contents

-
- Administrator

Learn how to create and edit fields codes to use in standard responses.

Related documentation:

-
-
-
-
-

Field codes are the most complex and powerful aspect of standard responses. With field codes, you can create standard responses that are automatically personalized when they are used.

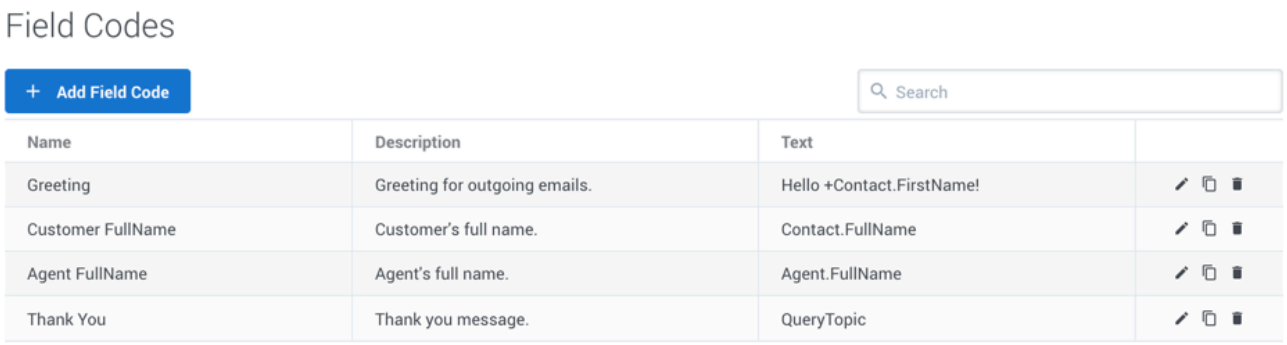
For example, you can use the field code `Contact.FirstName` in a response beginning `Dear Contact.FirstName`, which you send to dozens of recipients. In each message, `Contact.FirstName` is replaced by the first name of the addressee of the message (the contact) as listed in the Universal Contact Server database.

Field Codes Management enables you to create a wide range of field codes types. The codes can range from simple - similar to a Mail Merge word processor feature - to complex field codes that include multiple objects, formulas, and constants.

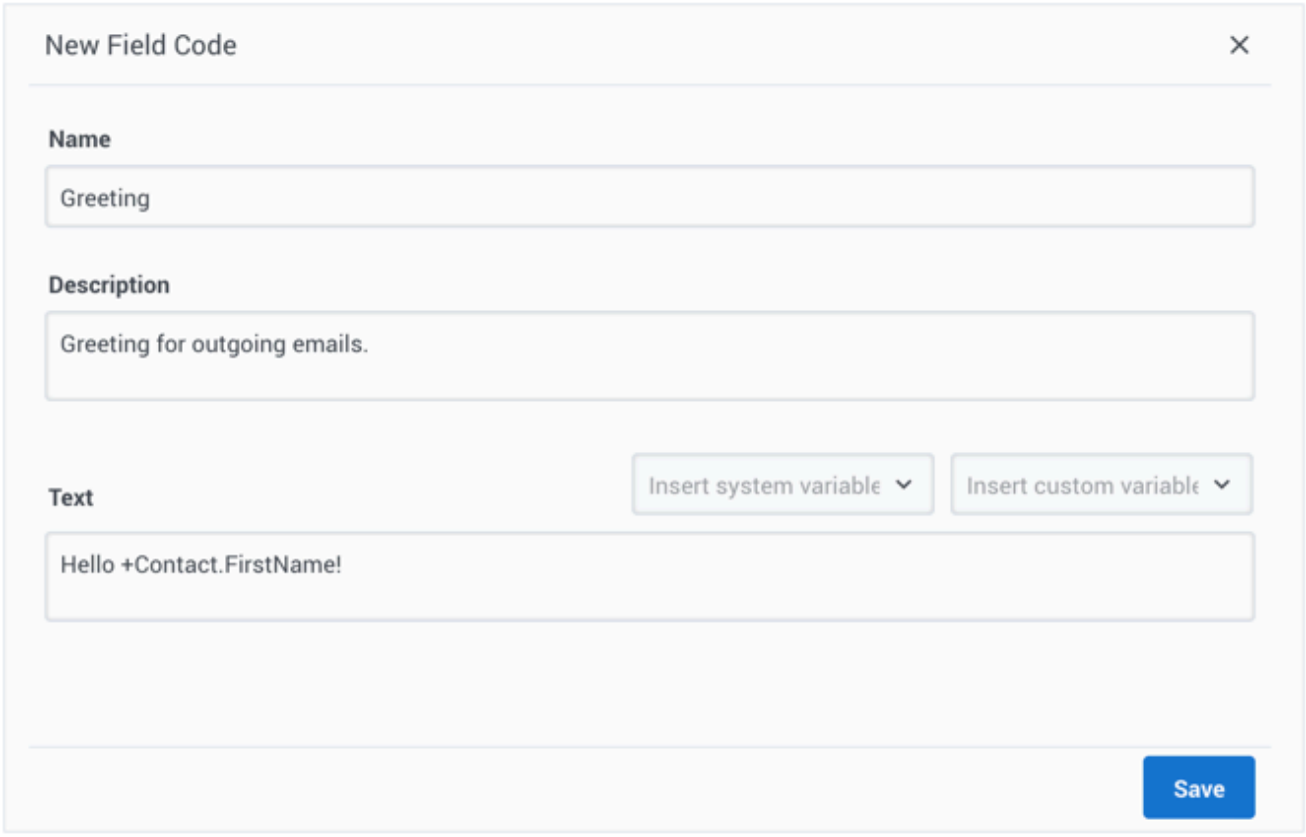
Once you create a field code, you can use it in multiple standard responses.

The interface for creating field codes is simple. Creating useful field codes requires a deeper understanding of how they are constructed. Create or edit a field code provides step-by-step instructions, with links to detailed reference information at the relevant places.

Access Field Codes Management under the **Digital Resources** menu.



Create or edit a field code



When you create a new field code, the **New Field Code** view opens on the right side of the page. To begin, give your field code a name and brief description.

> ## Important
>
> The field code name can consist of no more than 64 alphanumeric characters supported in UTF-8. For additional valid characters see Characters allowed in names. The following characters combinations are not allowed: "".

Choose system or custom variables to insert into the **Text** field, and add any other necessary text. See Use variables in field codes for details about variables and how to use them. If the custom variable you need is not in the list, you can create it using Custom Variables Management.

For detailed help with constructing the field code, see Use formulas in field codes.

See Examples for more about using complex field codes.

Use variables in field codes

The ability to access interaction data is perhaps the most frequent use of field codes. Although field code formulas can be complicated, many just retrieve a single piece of data, such as a contact's name.

You access Universal Contact Server data using predefined variables, called "system variables."

These variables access three predefined objects. Each object has a name and a set of properties. Let's use `Contact.FirstName` as an example. `Contact` is an object and `FirstName` is one of its properties. The system variable `Contact. FirstName` retrieves the value of the `FirstName` property of the `Contact` object.

Similarly, there is a system variable for each `object+property` pair. You can use the following objects in field codes:

- Agent object - Information about the agent, such as name and signature.
- Contact object - Contact details such as name, address, and phone number.
- Interaction object - Information about the interaction, such as created date, to and from address, and attached data.

Use your own data

It is possible to incorporate data that you keep external to Universal Contact Server into your standard responses (including automated responses). This data could include case numbers, account information, and so on. Remember that attached data always consists of key-value pairs.

Follow this two-step process to use external data:

1. Retrieve the external information and add it to the interaction as attached data. One place to do this is in a Designer application.
2. Now that you have attached the data to the interaction, use the `AttachedData` property of the Interaction object to access the data. The `AttachedData` property requires one argument, which is the key name. The result of the following formula is the value associated with the `OrderStatus` attached-data key: `Interaction.AttachedData("OrderStatus")`

Use formulas in field codes

In addition to system variables such as `Contact.FirstName`, field codes contain formulas. This section provides an outline of formula usage. Find details about these topics in the Field Codes Reference Guide.

Field code formulas are similar to formulas in other applications, such as Microsoft Excel.

A *formula* is a sequence of one or more operands (such as numbers and text strings), separated by operators (such as + and -).

For example, in the following formula, 2 and 3 are operands and + is an operator: 2 + 3

Operands can be values that do not change (constants), or values that vary based on the context. In the previous formula, all the operands are constants, so the formula always evaluates to `5`. The next formula evaluates to a different value for each agent who uses it: `Agent.Signature`

Field Code Syntax

To summarize field code syntax:

- Enclose alphabetic strings, whether constants in formulas or elsewhere in a field code, in double quotes.
- Numeric constants require no special treatment.
- Use special characters for some purposes. For example, for your field code to render with a line break, you cannot just type a carriage return. Instead, you must insert the expression `\n`. See the Escape Sequences table in the *Field Codes Reference Guide*.

HTML in Field Codes

With special configuration, field codes can contain HTML markup; for example, you could have a field code that uses a custom variable with a default value of:

```
Sam Agent

Acme Products

29 Exterior Blvd

Springfield, CX 09090
```

To enable this, you must use the Java property `-Dsrl-field-code-allow-html=true`, in one of the following ways:

- Add it to the `JavaArgs` section of `ContactServerDriver.ini`
- Add it as an argument to the startup command line in `contactServer.sh`.

Operator Precedence

If you use more than one operator in a formula, the order in which they are evaluated depends on their relative *precedence.* Higher precedence operators are evaluated first. For example, multiplication (`*`) has a higher precedence than addition (`+`), so that the formula below evaluates to `14`, not `20`:

```
2 + 3 * 4
```

You can use parentheses to override the default precedence. The formula below evaluates to `20`:

```
(2 + 3) * 4
```

For a complete list of operators and their relative precedence, see Operator precedence in the *Field Codes Reference Guide*.

Data Types

Operands of several different types may appear in formulas:

- Number
- String (text)
- Date/time
- Boolean (true/false)
- Object (Contact, Interaction, and Agent)

Each data type behaves differently in formulas, and the operators have different meanings when you use them with different data types. For example, the + operator means “add” when used with numbers, but “concatenate” (paste together) when used with strings. This formula evaluates to *Uncle Sam Wants You*:

```
"Uncle Sam " + "Wants You"
```

You can't use some operators with some data types at all. For example, you cannot use the multiplication (*) operator on two strings.

Genesys converts all formulas, regardless of their final data type, to strings before merging them into your standard response. This conversion follows a set of default rules that depend on the data type. For example, the default rules for numbers round them off to integers. This formula inserts 2 into your standard response, even though the real result is `2.25`:

```
9 / 4
```

You can use the `Text` function or format operator (:) to override the default formatting. Either of the following formulas inserts `2.25` into your standard response:

```
Text(9 / 4, "#.##")
```

```
(9 / 4):"#.##"
```

For a detailed list of data types and how you can use them, see Data types in the *Field Codes Reference Guide*.

Functions

When composing formulas, you can use many built-in functions. *Functions* are predefined formulas that perform calculations using values, called *arguments,* which you supply. To use a function, write its name, followed by an opening parenthesis, the arguments for the function separated by commas, and a closing parenthesis.

Function arguments can be of any data type, although individual functions can place restrictions on their arguments. Function arguments can be constants or formulas. The `Length` function, for example, takes a single string argument and returns its length in characters. This formula evaluates to 13:

```
Length("Hello, world!")
```

As another example, the `Date` function takes individual date components (year, month, day, and so on), and constructs a date/time value. The formula below evaluates to `1965-11-23 09:03:10`:

```
Date(1965, 11, 23, 9, 3, 10)
```

Functions can act as arguments to other functions. The `WeekdayName` function takes a single date/time argument and returns the day of the week as a string. The formula below evaluates to `Tuesday`:

```
WeekdayName(Date(1965, 11, 23, 9, 3, 10))
```

This formula evaluates to `7`:

```
Length(WeekdayName(Date(1965, 11, 23, 9, 3, 10)))
```

For detailed descriptions of all available functions, see the following topics in the *Field Codes Reference Guide*:

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

Using Objects

All object/property pairs are available in the system and custom variables menus.

Object properties can be of any data type. `Agent.FullName`, for example, is a string, but `Interaction.DateCreated` is a date/time.

The data type of an object property can even be another object. For example, `Contact.EmailAddresses` yields another object called a `ContactEmailAddressList`. You can access the properties of the resulting object by entering a period (.), followed by the property name, just as before. The formula below evaluates to the number of email addresses assigned to the contact:

```
Contact.EmailAddresses.Count
```

Some object properties require arguments just as functions do. For these properties, write the arguments, enclosed in parentheses after the property name, just as before.

For example, the ContactEmailAddressList object has a property named `Exists`. You can use this property test whether a contact has a particular email address. The data type of this property is Boolean (true/false), and it takes one argument, the email address to test.

```
Contact.EmailAddresses.Exists("samd@acme.com")
```

For detailed descriptions of all objects and their properties, see the following topics in the Field Codes Reference Guide:

- Agent object
- Contact object

- Interaction object

Examples

The following is an example of a complex field code:

```
If (Time() - Interaction.DateCreated > 14, “Please accept our apologies for not having replied sooner. ”, “”)
```

This field code inserts a tardiness apology if more than 14 days have elapsed since the interaction first entered the system. It uses the function `If`, which has these properties:

- Its syntax is `If (Boolean, TrueResult, FalseResult)`
- If `Boolean` evaluates to `True`, it returns the second argument.
- If `Boolean` evaluates to `False`, it returns the third argument.

In this example, the three arguments of `If` are as follows:

1. `Time() - Interaction.DateCreated > 14` A formula that returns `True` if the difference between the date created and the current system time is more than 14 days. (A mathematical operation on dates returns a result in days.)

1. `“Please accept our apologies for not having replied sooner.”` A text string apologizing for tardiness, inserted if the formula evaluates to `True`.
2. The null string: if the reply is not late (the formula evaluates to `False`), nothing is inserted in it.

Copy a field code

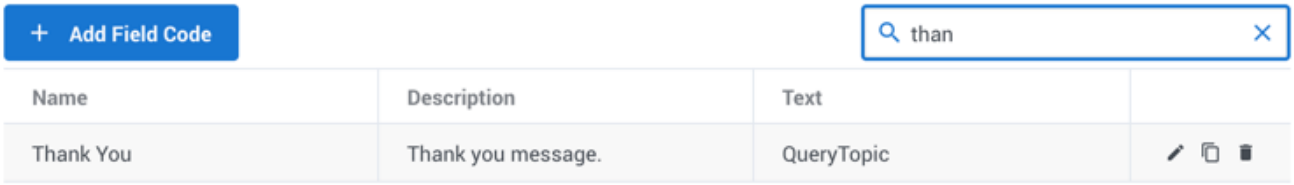
You can copy an existing field code with the **Copy** button ( ). This opens the **New Field Code** view with the copied information. The name of the field includes "*_N*", where "*N*" is the number of the copy. For example: Greeting_1, Greeting_2, and so on.

Delete a field code

To delete a field code, just click the delete icon ( ).

Search for a field code

To search for a field code, use the search box above the table. The results filter automatically based on the text you enter in the search box.



Characters allowed in names

Additional Characters Allowed in Object Names

Name	Character	Name	Character
Hyphen	-	Exclamation point	!
Number sign, pound	#	Dollar sign	$
Caret	^	Asterisk	*
Underscore	_	Curly brackets	{ }
Angle brackets		Period, full stop	.
Backslash	\	Parentheses	()
Question mark	?	Space	
At sign	@		