# Designer User's Guide

## Call Data Block

2/12/2026

## Contents

-

- Administrator

Use this block to read and write data that can be accessed from both inside and outside the application.

**Related documentation:**

-

The **Call Data** block enables you to read and write data that can be accessed from both inside and outside the application.

The data format consists of key-value pairs (KVP). For every **key** in the call data, you can associate a **value** with it. This value can be represented as an integer, character string, binary (boolean) type, or as a list of KVPs.

While data in application variables only exist within the application, information stored in call data can persist and be retrieved even after the application has stopped running. This allows an application to retrieve information about an interaction that was created before the application started running or preserve information about a customer or interaction that can be processed later by another application.

## Using this block

You can add the block to the **Self Service** or **Assisted Service** phase, but the block can behave differently depending on the phase in which it is used. These differences are noted where applicable, so be sure to review the information carefully when setting up the block.

In addition, the following recommendations and guidelines must be followed when using this block:

### Use fixed key names

Make sure to use fixed key names that won't change based on time, date, a varying ID of any kind, ANI, DNIS, or any other variables that can change between successive calls or digital interactions handled by the application. For example, avoid a key name like **myKeyName + ANI**, where **ANI** is the system variable that is automatically set by Designer.

**Warning**

Using dynamically changing key names is **not recommended**. They can cause significant performance issues and may be blocked in a future version of Designer without notice.

### Restricted variable names

There are certain variable names used by Designer that must not be updated by the **Call Data** block. See Restricted variable names for a list of these names.

### Secure and sensitive data

**DO NOT** attach sensitive data, such as personally identifiable information (PII) or secure variables, to user data in the **Call Data** block. Otherwise, this information is captured by platform logs and reported in Designer Analytics.

### Syntax requirements

When entering values, you must escape quote characters by using a preceding backslash. For example:

- **Correct:** `Joe\'s Pizza`
- **Incorrect:** `Joe's Pizza`

The **Call Data** block only accepts *string* type keys and values, so you do not need to enclose these values in single quotes.

## Read Data tab

Use the **Read Data** tab to specify keys to be read from Call Data and stored into application variables. You can toggle the key being read between a variable and a string.

## Edit Data tab

Use the **Edit Data** tab to specify key-value pairs to be written to the call data. There are some differences in how this block works, depending on whether it is added to the Self Service or Assisted Service phase. Make sure to review the details for each phase, below, so that you are aware of these differences.

### Use this data for async chat (does not apply to voice)

If enabled, this option allows the key-value pairs to be carried forward from the current chat session and used later in asynchronous chat sessions that start while the interaction is still being processed. For example, if a customer starts a chat session and is transferred to an agent, Designer can retrieve the KVPs from that session if the customer resumes activity in the chat window after the initial session timed out. This option is disabled by default.

## Self Service phase

If this block is in the **Self Service** phase, use the **Edit Data** tab to specify key-value pairs to be written to the call data (see an example). When used in **Self Service**, the **Call Data** block has the following restrictions:

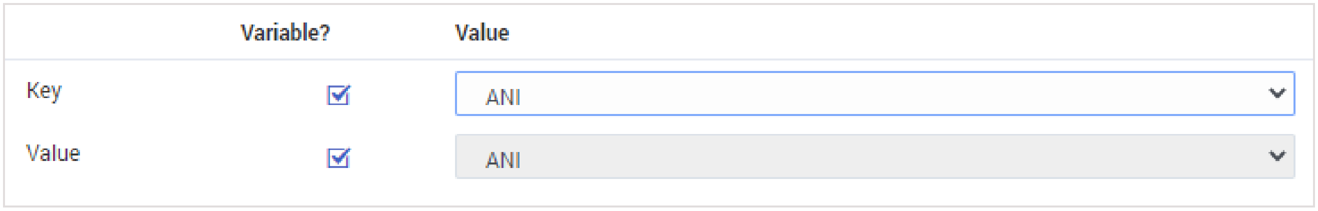
- Non-variable key names must follow standard JavaScript rules for variable names, even if the keys are not variables. For example, key names must only contain alphanumeric characters (no spaces) and not match any of the restricted variable names.
- Call data cannot be deleted. In the **Self Service** phase, the **Edit Data** tab does not have the **Remove** option, as is available in the **Assisted Service** phase.

When a key-value pair list is used as a value in the user data for a **Call Data** block in the **Self Service** phase, the user data is attached to the call, but the platform encodes any special characters that appear in the user data, such as single quotation marks ('), double-quotation marks ("), backslashes (\), commas (,), colons (:), semicolons (;), and so on.

You can specify *literal* or *variable* values (or a combination) for each KVP, using one of the following options:

### Variable Key + Variable Value

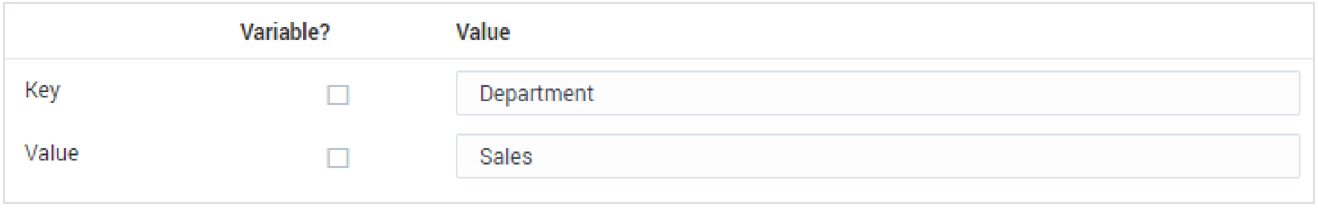
If you enable the variable checkbox for **Key**, the variable checkbox for **Value** is automatically selected and its dropdown menu is disabled (i.e. grayed out). Whichever variable you select for **Key** is also applied to **Value**. In this case, Designer will update the **Key** in the call data with the name of the variable and the **Value** with the value that is stored in it:



| | Variable? | Value |
|---|---|---|
| Key | ✓ | ANI |
| Value | ✓ | ANI |

### Literal Key + Literal Value

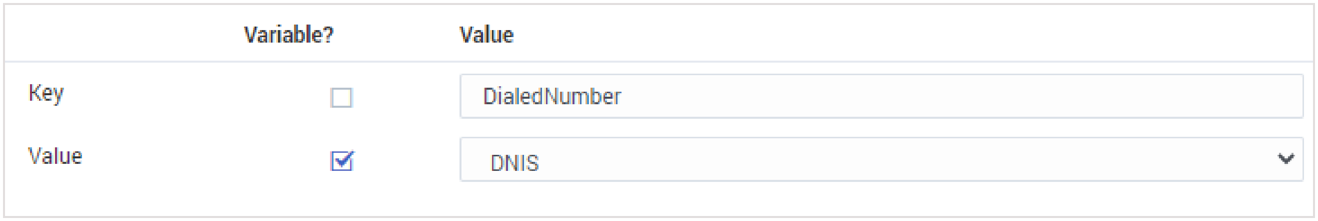
You can also enter literal values for both **Key** and **Value**. In this case, the entries are not required to match; each value is written to the call data as specified:



| | Variable? | Value |
|---|---|---|
| Key | ☐ | Department |
| Value | ☐ | Sales |

### Literal Key + Variable Value

If using a literal value for **Key** and specifying a variable for **Value**, make sure that the name of the variable you select does **NOT** match the entry provided for **Key**. Otherwise, the call data won't be updated correctly.



| | Variable? | Value |
|---|---|---|
| Key | ☐ | DialedNumber |
| Value | ✓ | DNIS |

### Assisted Service phase

If this block is in the **Assisted Service** phase, use the **Edit Data** tab to specify key-value pairs to be written to the call data, either by adding data with new keys (**Add/Update** operation), updating data for existing keys (**Add/Update** operation), or deleting data for existing keys (**Remove** operation):

- **Add/Update** - Add or update data with the specified key. This operation automatically adds the key-value if the specified key does not yet exist in the Call Data, or, if the provided key does exist in the Call Data, it automatically updates data for the key. You can toggle both the key and the value independently between a variable and a string.
- **Remove** - Provide the key for data you want to delete, which you can toggle between a variable and a string. At runtime, if the key you try to delete does not exist, nothing happens.

You can also add or remove a list of key-value pairs in a single operation with a JavaScript (JSON) object. For more information, see Using JSON objects to update or remove key-value pairs.

If you need to update a large amount of key-value pairs (i.e. 30 or more), see Adding or Removing Large Amounts of Key-Value Pairs.

## Advanced tab

If this block is in the **Assisted Service** phase, use the **Advanced** tab to add an extension as a key-value pair to the key. Click **Add Extension Data** to add an extension as a key-value pair to this block. The value type can be a string or integer.

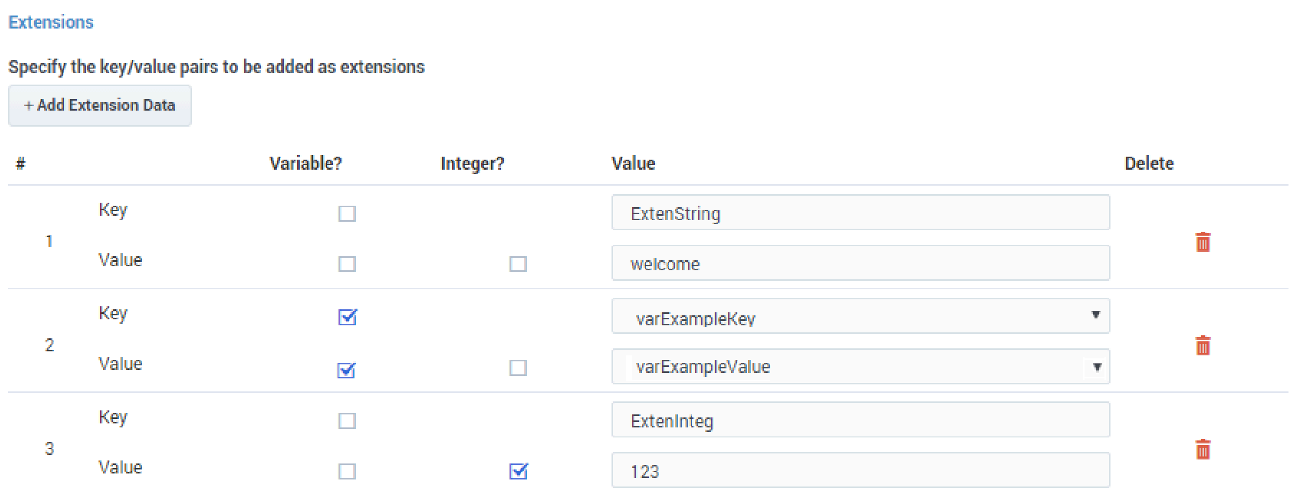
If you want to use a variable for the **Key** or **Value**, select the **Variable** checkbox and then select a variable from the drop-down menu. If the **Value** is an integer, select the **Integer** checkbox.

You do not need to enclose extension values in quotes. However, if the quote is part of the value, you must escape the quote character by using a preceding backslash (see Syntax requirements).

Important

Designer displays an error message if Extension Data is added, but the **Key** and **Value** settings are not defined.

This example shows a few different ways that key-value pairs can be added as extensions:


Extensions

Specify the key/value pairs to be added as extensions

+ Add Extension Data

| # | | Variable? | Integer? | Value | Delete |
|---|---|---|---|---|---|
| 1 | Key | ☐ | | ExtenString | 🗑 |
| | Value | ☐ | ☐ | welcome | |
| 2 | Key | ✓ | | varExampleKey | 🗑 |
| | Value | ✓ | ☐ | varExampleValue | |
| 3 | Key | ☐ | | ExtenInteg | 🗑 |
| | Value | ☐ | ✓ | 123 | |

## Examples

This section contains a few examples of how you can use the **Call Data** block.

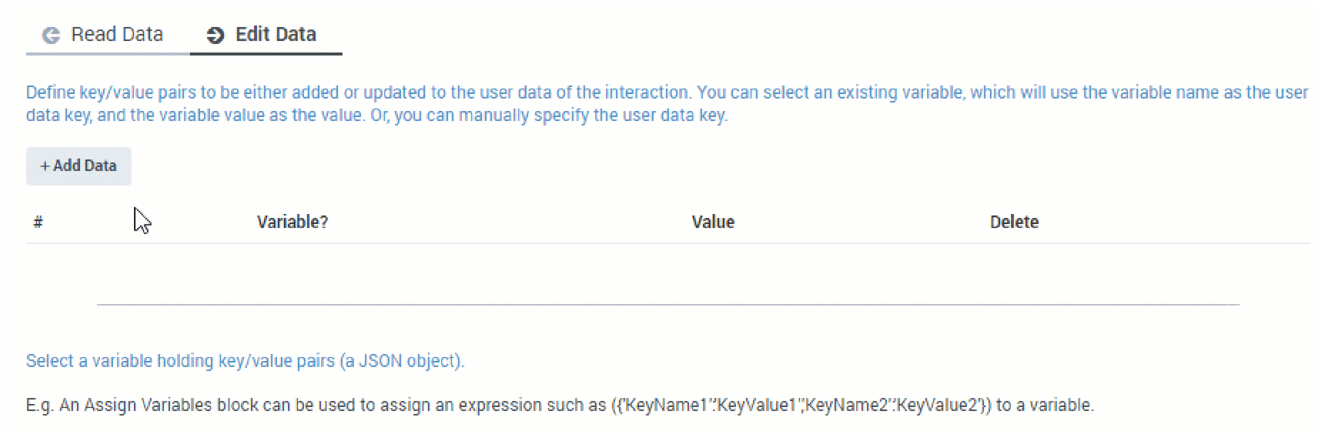
- Using the block in Self Service
- Using the block in Assisted Service
- Using JSON objects to add or remove key-value pairs
- Adding or removing large numbers of key-value pairs

### Self Service

In this example, we'll use the **Call Data** block to specify a key-value pair to be written to the call data. We've created a variable called **CustomerSegment** that Designer will use to store the segment detail about the customer (for example, **Bronze**) and we want to associate its value with the **segment** key in the call data.

On the **Edit Data** tab, we'll click **Add Data** to specify a new key-value pair. For the **Key** field, we'll enter **segment**. For the **Value**, we'll enable the **Variable** checkbox and select the **CustomerSegment** variable we created earlier.

Watch:



Define key/value pairs to be either added or updated to the user data of the interaction. You can select an existing variable, which will use the variable name as the user data key, and the variable value as the value. Or, you can manually specify the user data key.

| # | Variable? | Value | Delete |
|---|---|---|---|

Select a variable holding key/value pairs (a JSON object).

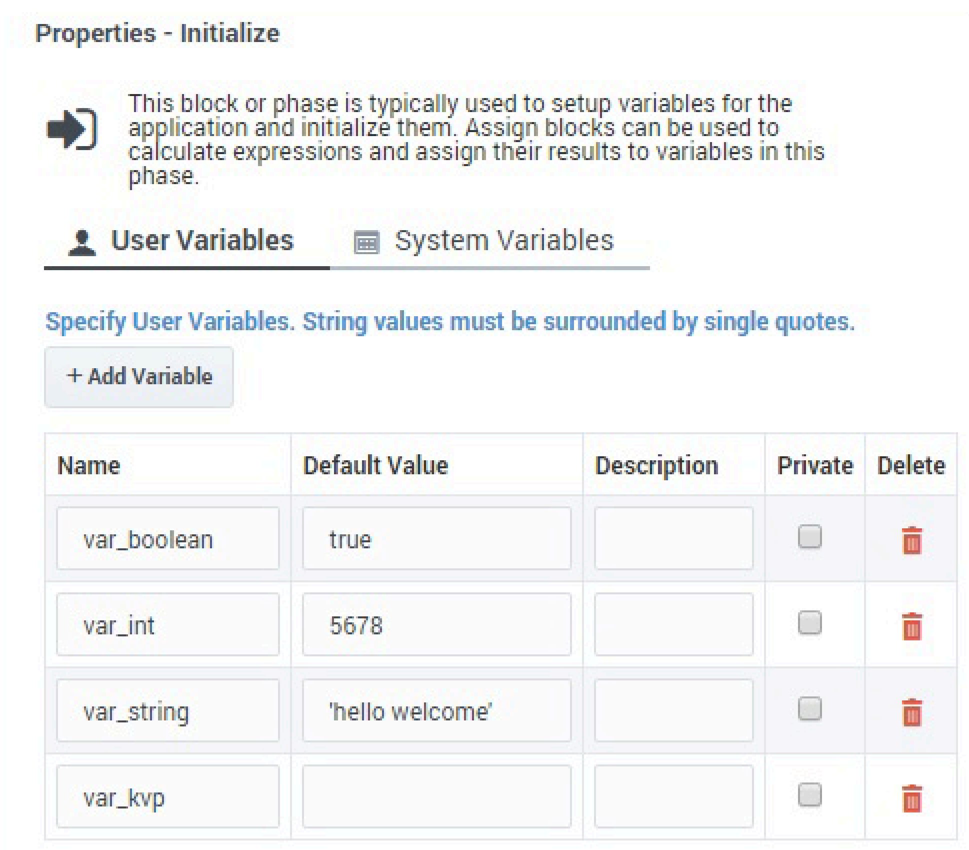
E.g. An Assign Variables block can be used to assign an expression such as ({'KeyName1':'KeyValue1','KeyName2':'KeyValue2'}) to a variable.

For more information about using the **Edit Data** tab in the Self Service phase, see Self Service phase.

## Assisted Service

This example shows how you can assign the value of a key-value pair in the **Call Data** block to a variable in the **Assisted Service** phase.

First, initialize your variables as various data types (integer, character string, binary [boolean], and so on), and create a variable for your key-value pair (in this example, we are using *var_kvp*):

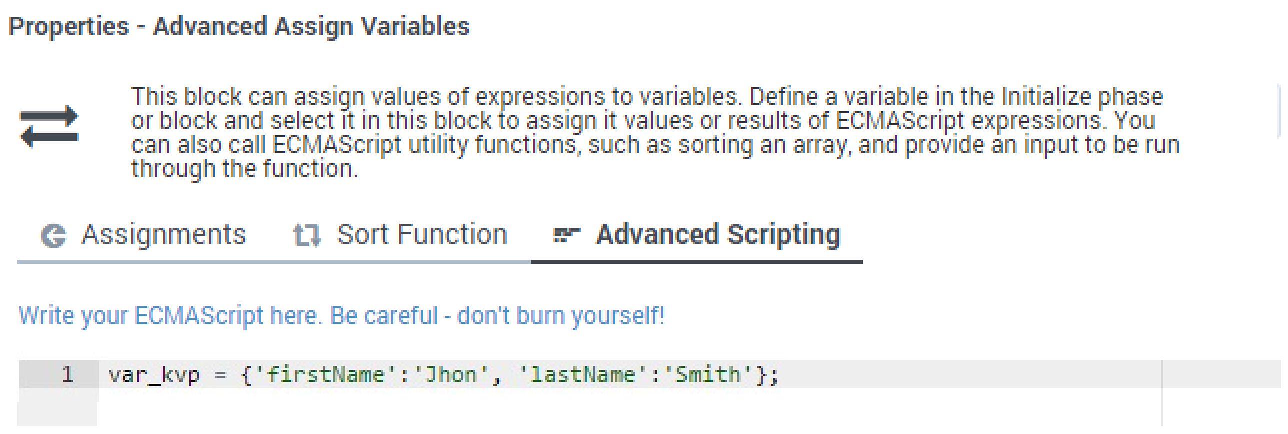

Properties - Initialize

This block or phase is typically used to setup variables for the application and initialize them. Assign blocks can be used to calculate expressions and assign their results to variables in this phase.

Specify User Variables. String values must be surrounded by single quotes.

| Name | Default Value | Description | Private | Delete |
|---|---|---|---|---|
| var_boolean | true | | ☐ | 🗑 |
| var_int | 5678 | | ☐ | 🗑 |
| var_string | 'hello welcome' | | ☐ | 🗑 |
| var_kvp | | | ☐ | 🗑 |

Next, initialize a variable as a list of key-value pairs using the Assign Variables block. In this example, we've done this using ECMAScript on the **Advanced Scripting** tab:

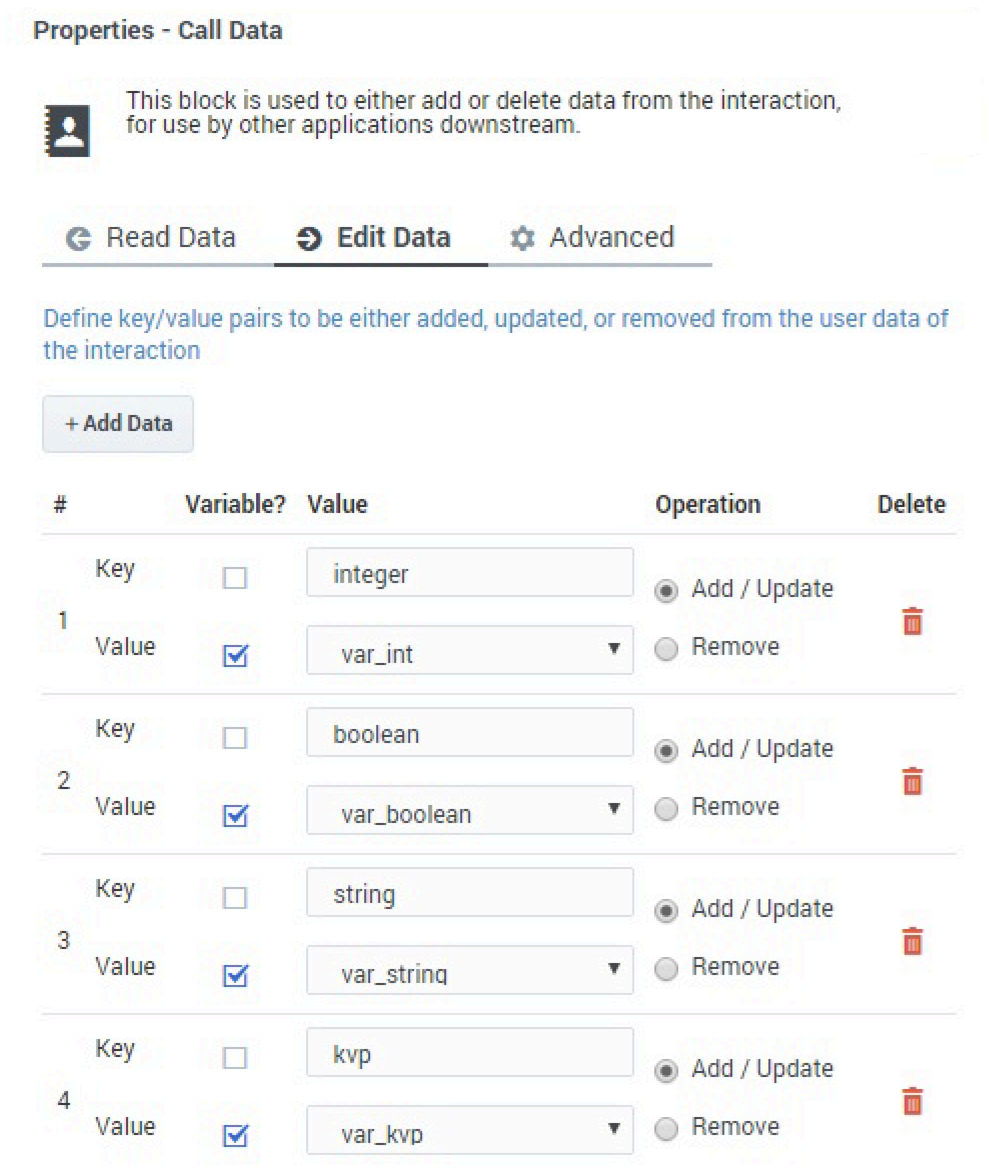

Properties - Advanced Assign Variables

This block can assign values of expressions to variables. Define a variable in the Initialize phase or block and select it in this block to assign it values or results of ECMAScript expressions. You can also call ECMAScript utility functions, such as sorting an array, and provide an input to be run through the function.

Write your ECMAScript here. Be careful - don't burn yourself!

```
var_kvp = {'firstName':'Jhon', 'lastName':'Smith'};
```

Finally, add the user data as a key-value pair in the Call Data block:


**Properties - Call Data**

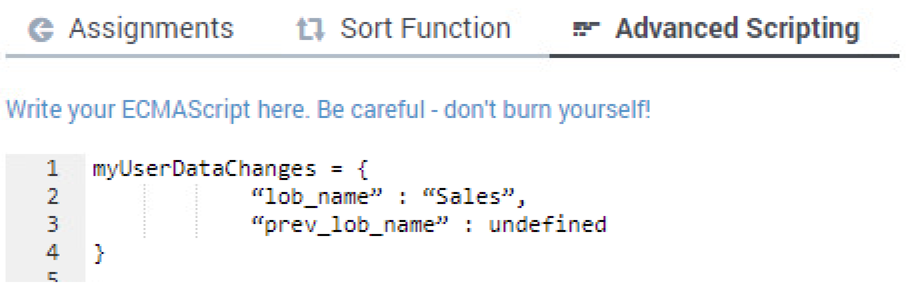
This block is used to either add or delete data from the interaction, for use by other applications downstream.

Define key/value pairs to be either added, updated, or removed from the user data of the interaction

| # | | Variable? | Value | Operation | Delete |
|---|---|---|---|---|---|
| 1 | Key | ☐ | integer | [✓ Add / Update] | 🗑 |
| | Value | ✓ | var_int | Remove | |
| 2 | Key | ☐ | boolean | [✓ Add / Update] | 🗑 |
| | Value | ✓ | var_boolean | Remove | |
| 3 | Key | ☐ | string | [✓ Add / Update] | 🗑 |
| | Value | ✓ | var_string | Remove | |
| 4 | Key | ☐ | kvp | [✓ Add / Update] | 🗑 |
| | Value | ✓ | var_kvp | Remove | |

## Using JSON objects to update or remove key values

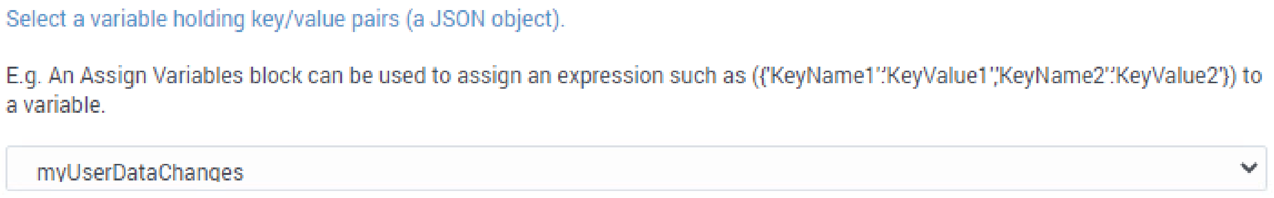
You can use the **Advanced Scripting** tab of the Assign Variables block to create a JSON object that can update or remove key values in a single operation. In this example, we've created a JSON object that will update the value for one key to **Sales** and delete the value for another by assigning it a value of **undefined**:



Write your ECMAScript here. Be careful - don't burn yourself!

```
myUserDataChanges = {
            "lob_name" : "Sales",
            "prev_lob_name" : undefined
}
```

You can then use the **Call Data** block to reference this JSON object and perform the operation. On the **Edit Data** tab, select the variable that contains the JSON object:



Select a variable holding key/value pairs (a JSON object).

E.g. An Assign Variables block can be used to assign an expression such as ({'KeyName1':'KeyValue1','KeyName2':'KeyValue2'}) to a variable.

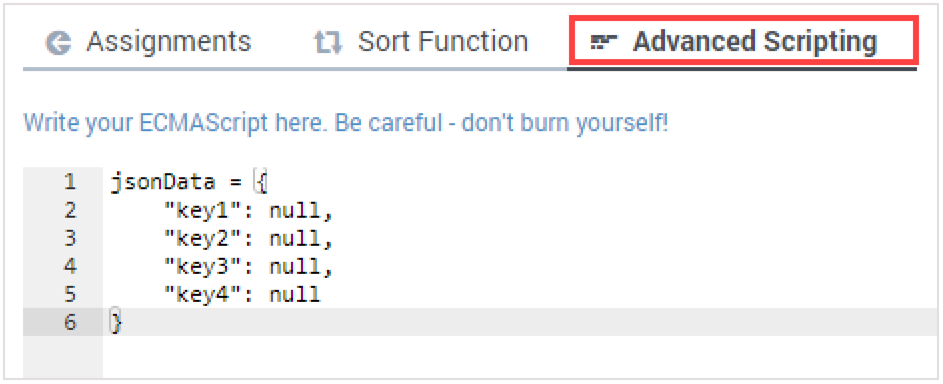
myUserDataChanges

This method can also be used to make bulk updates to a large amount of key-value pairs (i.e. 30 or more), as described in the next example.

## Adding or removing large numbers of key-value pairs

To add or delete several (i.e. 30 or more) key-value pairs, use the Advanced Scripting tab of an **Assign Variables** block to create a JSON object that lists each of the keys you want to modify.

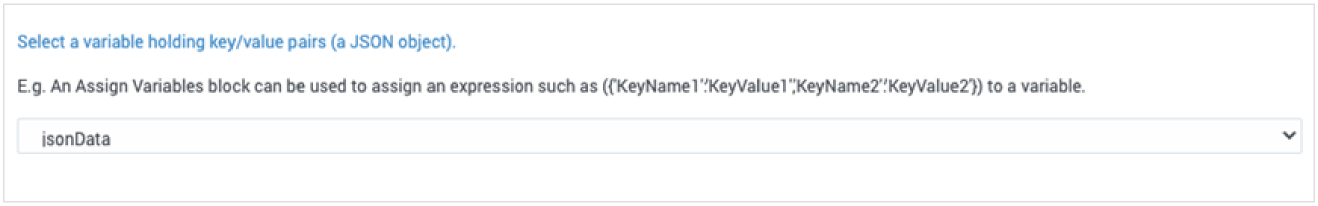
For example, to remove a large amount of key-value pairs, create a script that assigns each key a 'null' (or undefined) value. In this example, we've defined an object called **jsonData** and assigned each of its keys a value of **null**:



Write your ECMAScript here. Be careful - don't burn yourself!

```
jsonData = {
    "key1": null,
    "key2": null,
    "key3": null,
    "key4": null
}
```

If you wanted to add/update the keys, simply use the script to assign the desired values to each key. You can then reference this JSON object as a variable in the **Call Data** block to perform the desired operation:


Select a variable holding key/value pairs (a JSON object).

E.g. An Assign Variables block can be used to assign an expression such as ({'KeyName1':'KeyValue1','KeyName2':'KeyValue2'}) to a variable.

jsonData

Note that the operations are processed in smaller batches, to prevent too many requests from being processed at one time.

> **Tip**
> This method can be used in conjunction with the standard key-value pair operations described in Edit Data tab

## Restricted variable names

The following variable names are used by the Designer application and must not be updated by the **Call Data** block.

- **_CB_N_CUSTOMER_ABANDONED_WHILE_WAITING_FOR_AGENT**
- **_CB_SERVICE_ID**
- **_CB_T_CALLBACK_ACCEPTED**
- **_CB_T_CUSTOMER_CONNECTED**
- **_CB_T_SERVICE_START**
- **_COMPLETE_CS**
- **_SEND_FINAL_UE**
- **CustomerSegment**
- **EXECUTION_MODE**
- **GMS_UserData**
- **gvp-tenant-id**
- **gsw-ivr-profile-name**
- **GSYS_IVR**
- **GSYS_SystemApplicationDisposition**
- **IApplication**
- **IApplicationVersion**

- **IPurpose**
- **orsurl**
- **orssessionid**

## Warning

The addition, updating, and removal of Call Data do not happen instantaneously while the application is running. The application starts the operation to edit the Call Data, and then continues to the next block, without waiting for confirmation that the Call Data is successfully modified.

Thus, there is no guarantee that the "write" operation has finished—a subsequent "read" operation could potentially give you either an old value or an updated value on different runs through the application.

For this reason, Genesys recommends that you do not write a **key-value pair** and then attempt to read back the same **key** from the Call Data within an application. If you need to access a Call Data value that was already edited within the application, Genesys recommends that you use the corresponding application variable.