



This PDF is generated from authoritative online content, and is provided for convenience only. This PDF cannot be used for legal purposes. For authoritative understanding of what is and is not supported, always use the online content. To copy code samples, always use the online content.

## Genesys Designer Help

Call Data Block

# Call Data Block

## Contents

- **1 Call Data Block**
  - **1.1 Read Data tab**
  - **1.2 Edit Data tab**
  - **1.3 Advanced tab**
  - **1.4 Scenarios**
  - **1.5 Restricted Variable Names**
  - **1.6 Example**

## Important

The **Call Data** block has the following restrictions when used in the **Self Service** phase:

- Non-variable key names must follow standard JavaScript rules for variable names, even if the keys are not variables. For example, key names must:
  - Not contain spaces.
  - Only contain alphanumeric characters.
  - Not use a restricted variable name (see the [Restricted Variable Names](#) section, below).
- Call data cannot be deleted. The **Edit Data** tab does not have the **Remove** option, as is available in the **Assisted Service** phase.

You can use the **Call Data** block 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). That is, for every key in the call data, you can associate a value with it. This value can be an integer, character string, binary (boolean) type, or as a list of KVPs. (See an [example](#) of how to do this.)

**About key-value pair lists:** 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.

While data in application variables only exist within the application, information stored in Call Data can persist and be retrieved even after the application finishes running. This is useful for retrieving information about the call that was created before the application started running, or for saving some information about either a caller or a call to a database so that it can be processed later by another application.

## Warning

The addition, updating, and removal of Call Data do not happen instantaneously while the application is running. That is, 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.

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

Instead, 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.

### Important

- See the **Restricted Variable Names** section, below, for a list of variable names that must not be updated by the **Call Data** block.
- You must escape quote characters in values by using a preceding backslash. For example:
  - Incorrect: Joe's Pizza
  - Correct: 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


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.

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 a list of key-value pairs by selecting a variable that holds key-value pairs in a JavaScript object.

### Properties - Call Data

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

← Read Data  
 → **Edit Data**  
 ⚙️ Advanced

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

+ Add Data

#	Variable?	Value	Operation	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.

varJSONobject
▼

## 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.

### Important

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. For example:

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

### Important

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

### Example

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	<input type="checkbox"/>		ExtenString	
	Value	<input type="checkbox"/>	<input type="checkbox"/>	welcome	
2	Key	<input checked="" type="checkbox"/>		varExampleKey	
	Value	<input checked="" type="checkbox"/>	<input type="checkbox"/>	varExampleValue	
3	Key	<input type="checkbox"/>		ExtenInteg	
	Value	<input type="checkbox"/>	<input checked="" type="checkbox"/>	123	

### Scenarios

If you want to:

- Read the key **userphoneno** into application variable **phoneno** that you have previously created:
  - In the **Read** section, click **Add Key**.
  - Disable the **Variable?** check box.
  - In the **Key** field, enter userphoneno.
  - In the **Store in Variable** drop-down menu, select **phoneno**.

- Add the key **segment** with the value from the application variable **custsegment** that you have previously created:
  - In the **Edit** section, click **Add Data**.
  - Click **Add/Update**.
  - Disable the **Variable?** check box for **Key**.
  - In the **Value** field for **Key**, enter segment.
  - Enable the **Variable?** check box for **Value**.
  - In the **Value** drop-down menu for **Value**, select **custsegment**.

## 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**



## Example

This 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 KVP (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.

 **User Variables**     System Variables

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

[+ Add Variable](#)

Name	Default Value	Description	Private	Delete
var_boolean	true		<input type="checkbox"/>	
var_int	5678		<input type="checkbox"/>	
var_string	'hello welcome'		<input type="checkbox"/>	
var_kvp			<input type="checkbox"/>	

Next, initialize a variable as a list of kv-pairs using the **Assign Variables** block. This example shows this using an 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.

Assignments   Sort Function   **Advanced Scripting**

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

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

Finally, add the user data as a kv-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.

[← Read Data](#)   **[↔ Edit Data](#)**   [⚙️ Advanced](#)

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

+ Add Data

#	Variable?	Value	Operation	Delete
1	Key <input type="checkbox"/>	<input type="text" value="integer"/>	<input checked="" type="radio"/> Add / Update	
	Value <input checked="" type="checkbox"/>	<input type="text" value="var_int"/>	<input type="radio"/> Remove	
2	Key <input type="checkbox"/>	<input type="text" value="boolean"/>	<input checked="" type="radio"/> Add / Update	
	Value <input checked="" type="checkbox"/>	<input type="text" value="var_boolean"/>	<input type="radio"/> Remove	
3	Key <input type="checkbox"/>	<input type="text" value="string"/>	<input checked="" type="radio"/> Add / Update	
	Value <input checked="" type="checkbox"/>	<input type="text" value="var_string"/>	<input type="radio"/> Remove	
4	Key <input type="checkbox"/>	<input type="text" value="kvp"/>	<input checked="" type="radio"/> Add / Update	
	Value <input checked="" type="checkbox"/>	<input type="text" value="var_kvp"/>	<input type="radio"/> Remove	