

Using the Memory Adapter with Maps



This page explains the **Memory Adapter** in Bridge context. If you were looking for the same information regarding the [PAS Designer](#), refer to [Memory Adapter](#) in the Designer guide.

The Memory Adapter allows to store an entire hash map to memory. The hash map object as a whole can be managed with the Memory adapter the same way as any other objects. Additionally, the Memory adapter allows to add, change, retrieve, and remove elements from a stored hash map. Therefore, the Memory adapter provides an additional parameter **hashMapKey**. If this parameter is provided, the Memory adapter does not access the entire map, but tries to access the specified element of the map.

This can significantly increase the performance, if you e.g. want to cache some data you frequently need to lookup, in comparison to caching this data in a database.

Example File (Builder project E2E Action Language/Map):



<your example path>\E2E Action Language\map\uml\mapMemoryAdapter.xml

On this Page:

- [Storing a Map to Memory](#)
- [Retrieving Data from the Map](#)
 - [Related Error Codes](#)
- [Adding or Changing Data in the Map](#)
- [Removing an Element from the Map](#)
 - [Related Error Codes](#)
- [Removing the Whole Map](#)
- [Clearing the Memory](#)

Related Pages:

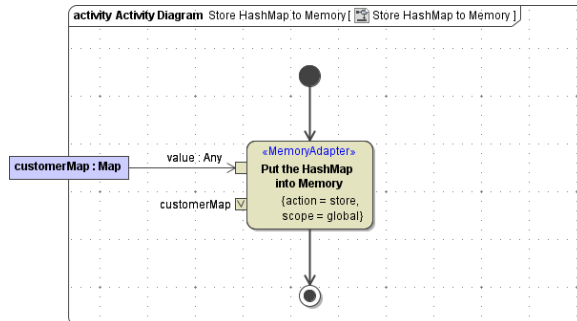
- [Map Operations](#)

Storing a Map to Memory

To store a map to memory, provide the Memory adapter with the following:

- the complete map in parameter **value**
- a **key**, that can be used to access the map in memory
- the Memory Adapter action **store**

Figure: Storing a Complete Hash Map to memory



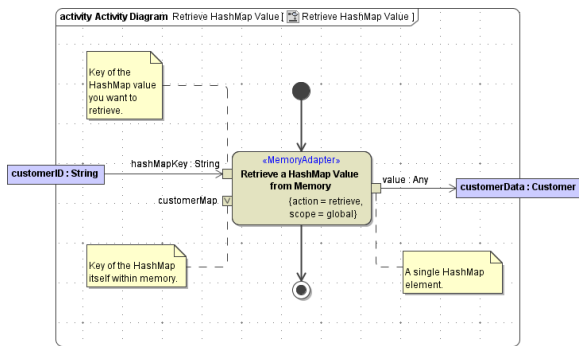
The figure above illustrates how to store map **customerMap** (containing some customer data) to memory. The map is stored using key "customerMap". If a map with the given key already exists in memory, it will be overwritten by the Memory Adapter. In this case, parameter **oldValue** returns the the complete previously stored hash map.

Retrieving Data from the Map

To retrieve an element from a map that has been stored to memory, provide the Memory adapter with the following:

- the **key** that has been used to store the map to memory ("customerMap" in the example below)
- the **hashMapKey** of the map element you want to retrieve
- the Memory Adapter action **retrieve**

Figure: Retrieving a Map Element from Memory



A hash map containing some customer data has been stored to memory using the key "customerMap" (see [Storing a Map to memory](#) further above). Individual map elements from this map can be accessed via their key **customerID**.

The Memory adapter returns a single map element containing the customer data.

If there is no map element with the given key, the Memory adapter throws an error.

Related Error Codes

Find a list of all persistent state error codes on [System Errors of the Memory Adapter](#).

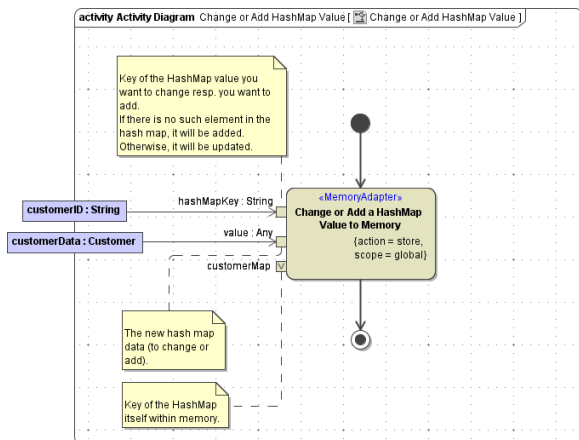
Error Code	Description
MEMADSM/9	The message is not stored here.

Adding or Changing Data in the Map

To add a new element to the map or to change map data in memory, provide the Memory adapter with the following:

- the **key** that has been used to store the map to memory ("customerMap" in the example below)
- the **hashMapKey** of the map element you want to retrieve
- the new **value** of the map element (stored in **customerData** in the example below)
- the Memory Adapter action **store**

Figure: Adding or Changing a Map Element of a Map in Memory



A hash map containing some customer data has been stored to memory using the key "customerMap" (see [Storing a Map to Memory](#) further above). Individual map elements from this map can be accessed via their key **customerID**.

The value of the new map element resp. the new value of the map element to be changed is provided in an object of type **Customer** via parameter **value** of type **Any**.

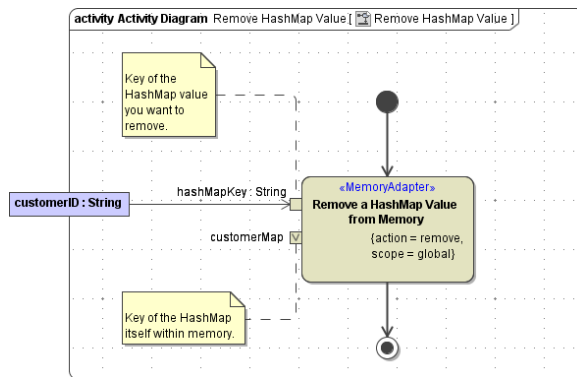
If there is **no map element with the given key**, the Memory adapter will add this element to the map.
 If the **map element with the given key is already existing** in the map, it will be updated to the provided value. In this case, parameter **oldValue** returns the complete previously stored hash map.

Removing an Element from the Map

To remove an element from a map that has been stored to memory, provide the Memory adapter with the following:

- the **key** that has been used to store the map to memory ("customerMap" in the example below)
- the **hashMapKey** of the map element you want to remove
- the Memory Adapter action **remove**

Figure: Removing a Map Element from a Map in Memory



A hash map containing some customer data has been stored to memory using the key "customerMap" (see [Storing a Map to Memory](#) further above). Individual map elements from this map can be removed via their key **customerID**. In this case, parameter **oldValue** returns the old map value. If there is no map element with the given key, the Memory adapter throws an error.

Related Error Codes

Find a list of all persistent state error codes on [System Errors of the Memory Adapter](#).

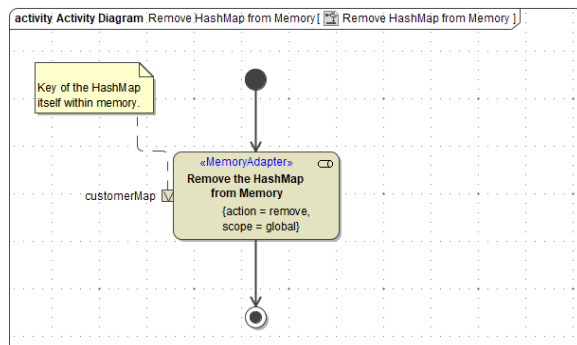
Error Code	Description
MEMADSM/12	The message is not stored here.

Removing the Whole Map

To remove the whole map from memory, provide the Memory adapter with the following:

- the **key** that has been used to store the map to memory ("customerMap" in the example below)
- the Memory Adapter action **remove**

Figure: Removing the Whole Map from Memory



The example above shows how the map that has been stored under key "customerMap" can be removed. In this case, parameter **oldValue** returns the the complete previously stored hash map. If there is no map with the given key, the Memory adapter throws an error(MEMADSM /12, see also [the list of system errors](#)).

Clearing the Memory

To clear the complete used memory, provide the Memory adapter with the following:

- the Memory Adapter action **clear**

Figure: Clearing the Memory

