3.x - Memory_Services

Memory caching and retrieval utility.

Syntax

```
Response = Memory_Services(Service, Param1, Param2, Param3, Param4, Param5, Param6, Param7, Param8, Param9, Param10)
```

Returns

The meaning of the response value depends on the service.

Parameters

Parameter	Description
Service	The name of the service being requested. **Required.**
Param1 - Param10	Generic parameters.

Remarks

This SRP FrameWorks utility service is designed to store small and large amounts of data in memory for quick retrieval. A very common use of `Memory_Services` is to store the results of other services. Thus, the beginning of each service would first check to see if a value already exists before going through the expense of running the entire service logic again. If the data being stored in `Memory_Services` needs to be refreshed after a short amount of time, the *GetValue* service can specify when this should expire.

Services

Service	Description
KeyExists	**Usage:** `Memory_Services('KeyExists', KeyID)` **Comments:** Returns a *True* or *False* depending on whether the Key ID exists. **Returns:** *True* if Key ID already exists in the SRP Hash Table, *False* if it does not exist.
GetValue	**Usage:** `Memory_Services('GetValue', KeyID, NotExpired, ExpirationDuration)` **Comments:** Returns the value pair stored in the SRP Hash Table for the current Key ID. If the *NotExpired* flag is set, the *ExpirationDuration* will be used to compare against the last time marker set for the current data. **Returns:** The value associated to the Key ID.
SetValue	**Usage:** `Memory_Services('SetValue', KeyID, Value)` **Comments:** Updates the value pair stored in the SRP Hash Table for the current Key ID. **Returns:** N/A

Remove Key	**Usage:** `Memory_Services('RemoveKey', KeyID)` **Comments:** Removes the Key ID, and its value pair, from the SRP Hash Table. **Returns:** N/A
CreateHashTable	**Usage:** `Memory_Services('CreateHashTable')` **Comments:** Creates the SRP Hash Table that the `Memory_Services` module will use to manage various Key ID and Value pairs. A check will first be made to see if the handle to the Hash Table already exists. If so then it will be released and a new Hash Table will be created. **Returns:** N/A
ReleaseHashTable	**Usage:** `Memory_Services('ReleaseHashTable')` **Comments:** Releases the SRP Hash Table handle. **Returns:** N/A
GetHandle	**Usage:** `Memory_Services('GetHandle')` **Comments:** Returns the handle to the SRP Hash Table used by `Memory_Services`. **Returns:** N/A

Param1 - Param10

The proper use of the generic arguments are defined in the definition of each service above.