Memory_Services

Memory caching and retrieval utility.

Syntax

```
Response = Memory_Services(@Service, @Params)
```

Returns

The meaning of the response value depends on the service.

Parameters

Parameter	Description
@Service	The name of the service being requested. Required.
@Params	Generic parameters. Refer to a specific service to determine the actual parameters used.

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, CacheName)
	Comments: Returns a <i>True</i> or <i>False</i> depending on whether the Key ID exists.
	Returns: <i>True</i> if Key ID already exists in the SRP Hash Table, <i>False</i> if it does not exist.
GetValue	Usage: Memory_Services('GetValue', KeyID, NotExpired, ExpirationDuration, CacheName)
	Comments: Returns the value pair stored in the SRP Hash Table for the current Key ID. If the <i>NotExpired</i> flag is set, the <i>ExpirationDuration</i> 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, CacheName)
	Comments: Updates the value pair stored in the SRP Hash Table for the current Key ID.
	Returns: N/A

lsValueEx pired	Usage: Memory_Services('IsValueExpired', KeyID, ExpirationDuration, ResetAge, CacheName)
	Comments: This relies upon the time marker set using the <i>SetValue</i> service. If this value has net yet been set then the value will be considered as expired.
	Returns: Returns a Boolean flag indicated whether the current value for the indicated KeyID has expired.
IsValueCu rrent	Usage: Memory_Services('IsValueCurrent', KeyID, ExpirationDuration, ResetAge, CacheName)
	Comments: This relies upon the time marker set using the <i>SetValue</i> service. If this value has net yet been set then the value will be considered as expired.
	Returns: Returns a Boolean flag indicated whether the current value for the indicated KeyID is still current.
RemoveK ey	Usage: Memory_Services('RemoveKey', KeyID, CacheName)
	Comments: Removes the Key ID, and its value pair, from the SRP Hash Table.
	Returns: N/A
CreateHas hTable	Usage: Memory_Services('CreateHashTable', CacheName)
	Comments: Creates the SRP Hash Table that the <u>Memory_Services</u> 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: Returns the handle to the newly created SRP Hash Table.
ReleaseHa shTable	Usage: Memory_Services('ReleaseHashTable', CacheName)
	Comments: Releases the SRP Hash Table handle. If <i>CacheName</i> is empty then the default handle is released.
	Returns: N/A
RemoveAl IHashTabl	Usage: Memory_Services('ReleaseHashTable')
es	Comments: Releases the all SRP Hash Table handles.
	Returns: N/A
GetHandle	Usage: Memory_Services('GetHandle', CacheName)
	Comments: Returns the handle to the SRP Hash Table used by Memory_Services.
	Returns: See comments.

Params

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