cloud.Start pKey, pURL

Initializes the library with your API key. The optional URL is the address of the cloud script - leave blank for the default.

Example

```
cloud.Start "abcdefabcdefabcdefabcdefabcdefab"
```

cloud.SetBlocking pState

Sets blocking mode to true or false. When blocking, you're script will pause until the request is complete. Non blocking allows your script to continue - a cloud.Response callback message will be sent when the cloud response is available.

Example

```
cloud.SetBlocking false
```

cloud.IsBlocking()

Returns the current blocking state : true or false.

Example

```
put cloud.IsBlocking() into tBlocking
```

cloud.Success()

Checks if the last cloud request was successful : true or false

Example

```
if cloud.Success() then ...
```

cloud.Error()
Returns the last error message.

Example

```
if not cloud.Success() then answer cloud.Error()
```

cloud.Stats()
Requests the account usage from the server. The reply is a comma separated list : Used (Kb) , Limit (Kb) , Percent.

Example

```
answer cloud.Stats()
```

cloud.Store pKey, pValue, pDeviceToken
Store a value on the server. pKey is the variable name and pValue is the value to be stored. If pDeviceToken is supplied, the value will be stored against that device token and can only be accessed by using the same token. Leave pDeviceToken blank for normal variable storage.

Example

```
# store a variable on the server
cloud.Store "EmergencyPhone", tPhone
```

```
# store a variable for the current device
cloud.Store "Credit", 250, tDeviceToken
```

cloud.Delete pKey, pDeviceToken
Delete a value from the server. pKey is the variable to be deleted. If pDeviceToken is supplied, only a value stored against that device token will be deleted.

Example

```
cloud.Delete "TodoList"
```

cloud.Fetch(pKey, pDeviceToken)
Fetches a variable from the server. If pDeviceToken is not blank then the device specific variable is returned.

Example

```
put cloud.Fetch("EmergencyContact") into tContact
```

cloud.FetchKeys(pDeviceToken)
Fetches the keys (variable names) of data stored on the server. If pDeviceToken is not blank then the keys stored against the token are returned. The keys are returned as a list - one per line.

Example

```
put cloud.FetchKeys() into tStoredKeys
repeat for each line tKey in tStoredKeys
  put cloud.Fetch(tKey) into gApplicationA[tKey]
end repeat
```

cloud.CheckScore(pKey, pScore, pMaxScores)
Checks the high score table identified by pKey. It takes pScore and determines if it is a new high score - based on the maximum number of scores to take into account : pMaxScores. Returns true for a new high score or false for a score too low.

Example

```
if cloud.CheckScore("HiScores", tUserScore, 10) then
  answer "New top 10 score!"
  .....
else
  answer "Better luck next time!"
  .....
end if
```

cloud.StoreScore pKey, pScore, pValue, pMaxScores
Stores a score in the high score table identified by pKey. pScore is the actual score and pValue is the label - such as the player's name or initials. pMaxScores specifies the maximum number of records to keep in the table : sending a value of 10 would delete any records after the top 10 once the table has been updated.

Example

```
# update the hi score table : keep the top 20 records
cloud.StoreScore "HiScores", tUserScore, tUserInitials, 20
```

cloud.DeleteScores pKey
Deletes the high score table identified by pKey.

Example

```
# delete the high score table for expert scores
cloud.DeleteScores "ExpertHiScores"
```

cloud.FetchScores(pKey)
Fetches the high scores stored in the table identified by pKey. The data returned is a list of high scores - one per line. Each line contains the high score and the player details - separated by a tab character.

Example

```
put cloud.FetchScores("HiScores") into field "HiScoreTable"
```

cloud.FetchScoreKeys()
Fetches the keys (identifiers) of high score tables stored on the server. The keys are returned as a list - one per line.

Example

```
put cloud.FetchScoreKeys() into field "HiScoreTables"
```

cloud.PushDeviceList(pOS)
Returns a list of device tokens - one per line - to receive push notifications. If pOS is specified ('iOS' or 'Android') then only devices for that platform are returned. if pOS is empty then devices for both platforms are returned.

Example

```
# fetch a list of devices for both iOS and Android
put cloud.PushDeviceList() into tDevices
```

cloud.PushRegister pOS, pToken, pValue
Register a device to receive (or stop receiving) push notifications. pOS if either 'iOS' or 'Android'.

pToken is the device token to use when sending the notification. Value is stored in the live flag in the database : A value of 'Y' enables push notifications and a value of 'N' disables notifications for the device.

Example

```
# register to receive push notifications
cloud.PushRegister "iOS", tDeviceToken, "Y"
```

cloud.PushBroadcast pPlatforms, pTitle, pAlert, pBadge, pSound, pPayload, pQueryKey, pQueryOp, pQueryVal
Broadcast a push notification to devices specified by pPlatform ('iOS' or 'Android'). If pPlatform is empty then devices for both platforms are targeted.

The next group of parameters are as follows;

- pTitle (Android only) - the notification title.
- pAlert - the alert text
- pBadge - a value to display next to the app icon
- pSound - notification sound to play
- pPayload - the payload to be sent to the app

The remaining three parameters target devices based on variables stored against the device tokens (experimental);

- pQueryKey - the key (name) of the stored variable
- pQueryOp - one of < , <= , = , != , <> , >= , > , LIKE , NOT LIKE
- pQueryVal - the value to match the variable to - using the query operator

Example

```
# send a push notification to all registered devices
cloud.PushBroadcast "", "News!", "News just in", 1, "", "app data....."
```

cloud.Resend()
Resends the last request.

Example

```
if not cloud.Success() then
  if cloud.Resend() then
    answer "OK on second attempt"
  else
    answer "Tried twice, but no dice"
  end if
end if
```

cloud.Response pResponseA
This message is sent to the object that made a non-blocking cloud request when the reply has been received from the server. pResponseA is an array containing the response data - most importantly;

- pResponseA["Success"] - true or false
- pResponseA["Data"] - the information requested

Example

```
on cloud.Response pResponseA
  if pResponseA["Success"] = true then
    # everything worked OK
    put pResponseA["Data"] into field "ServerResponse"
  else
    # something went wrong
    answer "An error occurred communicating with the server!" with "OK"
  end if
end cloud.Response1
```

cloud.SetPNCertificateIOS pCert, pDev
Upload the certificate required to send iOS push notifications. If you used the APNs assistant (available from http://www.splash21.com) to create the certificate files, then it's the AppCertKey.pem file.

```
-----BEGIN CERTIFICATE-----
MIIFiTCCBHGgAwIBAgIIBcuo8q47Z5EwDQYJKoZIhvcNAQEFBQAwgZYxCzAJBgNV
BAYTAlVTMRMwEQYDVQQKDApBcHBsZSBJbmMuMSwwKgYDVQQLDCNBcHBsZSBXb3Js
..........
```

```
..........
btuGvpaVwM/cpp3N0nUgCT/DZBK4Hawa4+NXMOwZtuUj5ehadH1LBkudroScMaCx
Now9rWTdxTGrwMBZ6JN/QwRpr6Rz1Hm6C0thB1s=
-----END CERTIFICATE-----
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAzf3dYLCumWMOWO2eWdz7MkF49JTj0+iLI8PwDiE0G6atvYPz
ZvxHqQsxjLZ20dbY0YviUr8Trq0A6pN6iItvMo2gIVySiiaSNXDP3BQCe6LJlw6j
..........
..........
nHND0le3MAylOqmXmt8eUVIF0XajEnZBNPZzA6bcRcoMuBOAPp7JZ+SLT5E7N88x
NdNkGCSMaIjP3RX+YUNl9Sn2I8dPvhMhiC9uXogSGow68fZZ8v1AWg==
-----END RSA PRIVATE KEY-----
```

If pDev is "Y" then the sandbox servers will be used to send push notifications. A value of "N" indicates that the live servers should be used.

Example

```
# upload PN certificate, using the sandbox service
cloud.SetPNCertificate the uCertificate of this card, "Y"
```

cloud.SetPNKeyAndroid pKey
Upload the API key created in the Google Developers Console for sending push notifications to android devices.

Example

```
# upload android GCM key
cloud.SetPNKeyAndroid "AlzaSyak...............bfgc"
```