Adaptive History List API for 5th Edition

Note: This API is not part of the public SDK. It can be found in the SDK API Plug-in.

API Purpose

The Adaptive History List API module provides APIs for applications to log items, request recommended lists, modify items, or to configure the Adaptive History List Engine. This API can be used by any application that requires the Adaptive History List functionality.

Example code

Header Files:

```
#include <ahleobserver.h>
#include <ahlegenericapi.h>
```

Link against:

```
LIBRARY    AHLE2CLIENT.lib
```

Capabilities Required:

```
CAPABILITY    ReadUserData WriteUserData
```

1) Derive your class from `MAHLEObserver`

2) Create an AHL object preferably in `constructL`

```
TPtrC databaseName = _L("SampleDb");    // database name for created client
iAhle = NewAHLEClientL(databaseName);   // iAhle is declared as MAHLEGenericAPI* iAhle in header file
iAhle->SetObserverL(this);
```

3) Log a new item in the AHL database
If the item does not exist, a new entry is created in the list. Otherwise, the weight of the already existing item is updated according to the algorithm.

```
TPtrC item = _L("Item");    // key value for item
TPtrC name = _L("Name");    // name of item (descriptive inforamtion about item)
TReal32 weight = 2.0f;     // base weight of item
TInt err = iAhle->NewAccess(item, name, weight);
//Name and weight can be omitted. Weight can be used to categorize items:
//more important items should have higher base weight value.
```

4) Rename an item

```
TPtrC item = _L("Item");    // key value for item
TPtrC newName = _L("New name");    // new item's name
TInt err = iAhle->RenameL(item, newName);
```

5) Delete one item from the AHL database

```
TPtrC item = _L("Item");    // key value for item
TInt err = iAhle->Remove(item);
```

6) Delete several items matching a specified string from the AHL database

```
TPtrC match = _L("It");    // match for item's key value
TInt err = iAhle->RemoveMatching(match);
```

7) Clear the database (delete all items)

```
TInt err = iAhle->Clear();
```

8) Implement the method `AdaptiveListChangedL()`
It is a pure virtual method defined in `MAHLEObserver`, called by the framework whenever the AHL database is changed.

```
void AdaptiveListChanged(TInt aError)
```


(C) Copyright Nokia 2013. All rights reserved.

```
  {
  if(!aError)
    {
    DoAdaptiveListChangedL();
    }
  }
```

9) Get an adaptive list of items.

```
void DoAdaptiveListChangedL()
  {
  RFs myFileSession;
  RFile myFileObject;
  CleanupClosePushL(myFileSession);
  CleanupClosePushL(myFileObject);
  CDesCArrayFlat* items = new (ELeave) CDesCArrayFlat(20);
  CleanupStack::PushL(items);
  CDesCArrayFlat* names = new (ELeave) CDesCArrayFlat(20);
  CleanupStack::PushL(names);
  TUint size = 3;
  TPtrC match = _L("A");
  myFileSession.Connect();
  myFileObject.Replace(myFileSession,_L("C:\\History.txt"), EFileWrite);
  TInt err = iAhle->AdaptiveListL(*items, *names, size, match);
  TBuf8<500> tempBuffer;
  for(TInt i=0;i<names->Count();i++)
    {
    tempBuffer.Append(names->MdcaPoint(i));
    tempBuffer.Append(_L8("-"));
    tempBuffer.Append(items->MdcaPoint(i));
    tempBuffer.Append(_L8("\n"));
    }
  //logs the contents of the buffer into a file
  myFileObject.Write(tempBuffer);
  tempBuffer.Zero();
  CleanupStack::PopAndDestroy(4); // names, items, myFileObject, myFileSession
  }
```


(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.