AIW Criteria API

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

AIW Criteria API is used to offer the definitions and classes for utilizing the AIW criteria items and interests. An interest is an array of criteria items, and it defines what AIW services the consumer application is interested in using. The AIW services are, in turn, offered by the AIW service providers.

Use cases

The most important use cases of AIW Criteria API are the following:

Defining the consumer's interest in a resource file

Defining the consumer's interest dynamically

Making asynchronous service calls.

Example code

Defining the consumer's interest in a resource file

The consumer's interest is usually defined in a resource file. An example case containing menu and base service interests is shown below: Header files

```
#include <AiwCommon.rh>
```

```
RESOURCE AIW_INTEREST r_aiwexample_menuinterest
    {
    items =
        {
        AIW_CRITERIA_ITEM
            {
            id = EAIWExampleHelpPlaceholder;
            serviceCmd = KAiwCmdHelp;
            contentType = "*";
            serviceClass = KAiwClassMenu;
            }
        };
    }
```

```
RESOURCE AIW_INTEREST r_aiwexample_baseinterest
    {
    items =
        {
        AIW_CRITERIA_ITEM
            {
            id = EAIWExampleBaseServiceId;
            serviceCmd = KAiwCmdMnShowMap;
            contentType = "application/x-landmark";
            serviceClass = KAiwClassBase;
            }
        };
    }
```

In this example, only one criteria item is defined per each interest, but there could be more as well, separated by comma.

The first criteria item refers to a menu service, i.e. the corresponding AIW provider(s) are supposed to add menu item(s) to the consumer application's menu. The second one refers to a base service. Using base service commands do not require any menus. The id enumerations should be defined in the consumer application's HRH file.

The menu items for menu services are defined in a MENU_PANE resource. An example is given below:

```
RESOURCE MENU_PANE r_aiwexample_menu
    {
    items =
        {
        MENU_ITEM
            {
            command = EAIWExampleHelpPlaceholder;
            txt = "Help submenu";
            cascade = AIW_INTELLIGENT_CASCADE_ID | AIW_LOCK_SUBMENU_TITLE;
            },
        MENU_ITEM
            {
            command = EAknSoftkeyExit;
            txt = "Exit";
```


(C) Copyright Nokia 2013. All rights reserved.

```
            }
        };
    }
```

In this example, the first item is an AIW menu item and the second is a normal menu item. AIW_INTELLIGENT_CASCADE_ID tells that the AIW framework should place the AIW menu items in a submenu, if the provider(s) offer several AIW menu items. If there is only menu item available, it is located at the main level.

AIW_LOCK_SUBMENU_TITLE tells the AIW framework that the consumer defined string "Help submenu" should be used if the submenu exists. This means that if some provider suggests a submenu title, it will be ignored.

Defining the consumer's interest dynamically

The interest may also be defined dynamically. See example below:

```
// Create AIW service handler
CAiwServiceHandler* serviceHandler = CAiwServiceHandler::NewLC();

// Create AIW interest
RCriteriaArray interest;
CleanupClosePushL(interest);
_LIT8(KContentTypeLandmark, "application/x-landmark");
CAiwCriteriaItem* criteria = CAiwCriteriaItem::NewLC(KAiwCmdMnShowMap,
                                    KAiwCmdMnShowMap, KContentTypeLandmark);
// We are using a base service.
TUid base;
base.iUid = KAiwClassBase;
criteria->SetServiceClass(base);

User::LeaveIfError(interest.Append(criteria));

// Attach the interest to the AIW framework.
serviceHandler->AttachL(interest);

...
// Execute AIW commands etc.
...

// Pop and destroy when not any more needed.
CleanupStack::PopAndDestroy(3); // criteria, interest, servicehandler
```

In this example, the KAiwCmdMnShowMap constant is used also as a criteria item id. This approach is also OK, the consumer can decide the id it uses (and for base services the criteria item id is quite meaningless).

Making asynchronous service calls

Some AIW service providers might support asynchronous service calls. This might be the case e.g. in such situations, where the processing takes so much time that the provider cannot return immediately.

In such case the consumer should derive from class MAiwNotifyCallback and implement its MAiwNotifyCallback::HandleNotifyL() method. When the consumer calls either the CAiwServiceHandler::ExecuteMenuCmdL() or CAiwServiceHandler::ExecuteServiceCmdL(), it should set the aCallback pointer to the M class in question and also set the aCmdOptions to KAiwOptASyncronous. The service provider may then call that callback method with e.g. KAiwEventCompleted when finished. See AiwCommon.hrh for a list of possible event codes.

An example how to derive from MAiwNotifyCallback is shown below: Header files

```
#include <AiwCommon.rh>
```

```
class CMyConsumerApp : public MAiwNotifyCallback
    {
    ...
    public:
        // From MAiwNotifyCallback
        virtual TInt HandleNotifyL(
            TInt aCmdId,
            TInt aEventId,
            CAiwGenericParamList& aEventParamList,
            const CAiwGenericParamList& aInParamList);
    ...
```

The implementation for HandleNotifyL() could look like the following:

```
TInt CMyConsumerApp::HandleNotifyL(
    TInt /*aCmdId*/,
    TInt aEventId,
    CAiwGenericParamList& /*aEventParamList*/,
    const CAiwGenericParamList& /*aInParamList*/)
    {
    // Service command (aCmdId) can be checked here, if necessary.

    // Check the event code.
    switch (aEventId)
        {
        case KAiwEventCanceled:
            {
            // Handle cancellation...
            break;
            }
        case KAiwEventCompleted:
            {
            // Handle completion...
            break;
            }
```


(C) Copyright Nokia 2013. All rights reserved.

```
        ...
        // Other event codes can be handled here as well.
        ...
        default:
            break;
        }

    return KErrNone;
    }
```

Asynchronous command handing can be requested as shown below:

```
void CMyConsumerApp::HandleCommandL(TInt aCommand)
    {
    ...
    // Execute AIW menu service command asynchronously. Remember to check from
    // the provider documentation whether it supports asynchronous command handling.
    // No input and output paramters are used in this example. Check from the
    // provider documentation whether those are required.
    iServiceHandler->ExecuteMenuCmdL(
        aCommand,                           // The menu command
        iServiceHandler->InParamListL(),    // No input parameters used
        iServiceHandler->OutParamListL(),   // No output parameters used
        KAiwOptASyncronous,                 // Asynchronous command handling requested
        this );                             // MAiwNotifyCallback pointer
    ...
```

Example project

File:AIWConsumerBasics.zip

Known issues

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.