AIW Service Handler API

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

The purpose of the AIW Service Handler API is to offer the core functionality of the Application Interworking framework to the AIW consumer applications. This is done by using a Service Handler object, which is the core of the AIW framework.

Use cases

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

Creating a Service Handler instance

Attaching interests

```
Base services
Menu services
```

Initializing menu pane

Executing service commands

Base service commands

Menu service commands

Deleting the Service Handler instance

Example code

Creating a Service Handler instance For using AIW Service Handler API, the consumer application needs first to create a Service Handler instance.

```
// Create a service handler instance.
iServiceHandler = CAiwServiceHandler::NewL();
```

Attaching interests

When the Service Handler instance exists, the consumer application's interest must be attached to it before AIW service commands can be executed. Attach needs usually be done only once, and a good place for it is the consumer application's ConstructL() method.

Base services

Base service interests can be attached by using CAiwServiceHandler::AttachL().

```
// Attach base service interests.
iServiceHandler->AttachL(R_AIWEXAMPLE_BASEINTEREST);
```

The interest is defined in a resource file, see AIW Criteria API for details.

Menu services

Menu service interests can be attached by using CAiwServiceHandler::AttachMenuL().

```
iServiceHandler->AttachMenuL(R_AIWEXAMPLE_MENU, R_AIWEXAMPLE_MENUINTEREST);
```

A menu pane and an interest containing menu related criteria items need to be defined in a resource file, see AIW Criteria API for details.

Initializing Menu pane

Menu pane containing AIW menu items needs to be initialized. This is done in consumer application's DynInitMenuPaneL() method. See example below:

```
void CAIWExampleAppUi::DynInitMenuPaneL( TInt aResourceId, CEikMenuPane *aMenuPane )
```


(C) Copyright Nokia 2013. All rights reserved.

```
    {
    // First, offer menu pane to AIW framework. It might be the case, that the
    // user is opening an AIW submenu. In this case, the AIW handles the menu.
    if ( iServiceHandler->HandleSubmenuL( *aMenuPane ) )
        {
        return;
        }

    // Add your normal (non-AIW) menu initialisation code here...

    // Let AIW provider add its menu items to the menu.
    iServiceHandler->InitializeMenuPaneL(
        *aMenuPane,
        aResourceId,
        EAIWExampleCmdLast,
        iServiceHandler->InParamListL());
    }
```

In this case, the menu may contain an AIW submenu (i.e. a submenu containing only AIW menu items). The submenu is handled by the AIW framework, so if CAiwServiceHandler::HandleSubmenuL() returns ETrue, nothing needs to be done.

The actual menu pane initialization is done by CAiwServiceHandler::InitializeMenuPaneL(). Note that the EAIWExampleCmdLast should be defined as the last enumeration value, the Service Handler uses it as the basis of the menu command ids it generates.

Executing service commands

AIW service commands can be executed by calling CAiwServiceHandler::ExecuteServiceCmdL() for base services, and CAiwServiceHandler::ExecuteMenuCmdL() for menu services. The methods are blocking by default. However, it is also possible that a provider works asynchronously. In that case, the consumer can implement a callback method, which is called when the provider finishes, for example. See AIW Criteria API for more details of asynchronous service calls.

Base service commands

Base service commands can be executed by calling CAiwServiceHandler::ExecuteServiceCmdL(). An example is shown below (see AIW Generic Parameter API for how to set up input parameters):

```
iServiceHandler->ExecuteServiceCmdL(
    KAiwCmdMnShowMap,                  // The service command.
    inParamList,                       // Input parameter list.
    iServiceHandler->OutParamListL(),  // No output parameters used.
    0,                                 // No options used.
    NULL);                             // No need for callback
);
```

Menu service commands

The consumer application's "AppUI" class has a callback method HandleCommandL(TInt aCommand). When it is called, the consumer application should first try to identify if the command is a normal menu command. If the command is not recognized, it should be forwarded to the AIW framework. This is usually done in the default branch of the switch statement.

```
void CAIWExampleAppUi::HandleCommandL(TInt aCommand)
    {
    switch(aCommand)
        {
        case ESomeNonAIWCommand:
            {
            // Execute command.
            // ...
            break;
            }
        case EEikCmdExit:
            {
            Exit();
            break;
            }
        default:
            {
            // Forward the command id to AIW, i.e. execute AIW menu
            // service command.
            iServiceHandler->ExecuteMenuCmdL(
                aCommand,                          // No input parameters
                iServiceHandler->InParamListL()    // No output parameters
                iServiceHandler->OutParamListL(),  // No options used.
                0,                                 // No need for callback
                NULL);
            break;
            }
        }
    }
```

Deleting the Service Handler instance

The Service Handler instance needs to be deleted when it is not used anymore. Usually this is done in the destructor of the consumer application.

```
// Delete the service handler instance.
delete iServiceHandler;
```


(C) Copyright Nokia 2013. All rights reserved.

Example project

File:AIWConsumer.zip

File:Emergency Caller.zip

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.