Browser Control Symbian API - Cloning the connection for Download Manager

Description

When starting the download from the browser control application, the Download Manager server starts up and a connection is created and opened from the client process (BrCtlSampleApp).

On request to complete, the Download Manager tries to use the existing connection in `CHttpConnHandler::ConnectL()` to get callbacks from the network layer:

```
// This will try to clone the opened connection
User::LeaveIfError(iConnection.Open(iClient->Engine()->SocketServ(), connName));
```

but fails to clone the existing connection in the Download Manager server process and leaves with -46 (`KErrPermissionDenied`) error without any callbacks. The download can, however, be started using `StartL`.

If the paused download is resumed from the application, Download Manager will try to use the existing connection in the same way but again, fails to clone the connection. No callbacks are observed, thus preventing the paused download to resume.

Solution

In brief, the solution to this problem would be rewriting the code so that the connection could be cloned from the Download Manager server process.

Basically it is the responsibility of the donor process (the process that opened the connection before it is cloned) to mark a connection to be a clonable one. Unless this is done, another process (the receiver) cannot clone the connection. In this case the Donor is `BrCtlSampleApp` and the Receiver is the Download Manager server.

So if the donor (`BrCtlSampleApp`) gives the control to clone the existing connection, the Download Manager server process can open/clone the existing connection successfully and can get callbacks from the network layer. The HTTP request made by the Download Manager server can be completed and can receive packets also after resuming.

The Donor application can give permission to clone the connection opened by it by just using the `RConnection::Control()` API as follows:

```
// Enable connection cloning
LIT_SECURITY_POLICY_C2(KProcPolicy1,
                       ECapabilityNetworkServices,
                       ECapabilityNetworkControl);
TSecurityPolicyBuf secPolBuf;
secPolBuf().Set(KProcPolicy1().Package());
iConnection.Control(KCOLConnection, KCoEnableCloneOpen, secPolBuf);
```

Note that before platform security, any application could clone and then use the connection that was opened by another process. A process could clone a connection if it knew the connection's unique global name, obtained by `RConnection::Name()`.

However, platform security requires the donor process to mark a connection as one that can be cloned. Unless this is done, another process (the receiver) cannot clone the connection.

In addition to simply marking a connection, the donor process can specify the capabilities needed by the receiver in order to clone the connection.

For more information, see the example application File:BrCtlSampleApp.zip.

 (C) Copyright Nokia 2014. All rights reserved.

 (C) Copyright Nokia 2014. All rights reserved.