

Contents

- [1 Introduction](#)
- [2 How to Connect](#)
- [3 How to request for folder listing](#)
- [4 How to request for folder listing](#)
- [5 How to handle folder listing](#)
- [6 Example Applications](#)

Introduction

The purpose of this document is to show how we can use Obex protocol to browse folder over Bluetooth link. Due to security reason we can't browse all folders (like any file browser in local devices). We connect the remote device to FTP service with a particular port number. So before using obex, we should discover the remote FTP services and it's port number. We can use normal service discovery protocol for these purposes.

How to Connect

First we need to connect to FTP service.

```

/*
 * @param aRemoteDevice remote Bluetooth device, it's folder is going to be browsed
 * @param aRemotePort remote port number by which it is going to be connected
 */

void CHemeBTBrowserObexClient::ConstructL( const TBTDevAddr& aRemoteDevice,
                                           const TUint aRemotePort)
{
    TObexBluetoothProtocolInfo info;
    info.iTransport = KBTSProtocol;
    info.iAddr.SetBTAddr( aRemoteDevice );
    info.iAddr.SetPort( aRemotePort );

    TObexProtocolPolicy obexProtocolPolicy;
    obexProtocolPolicy.SetReceiveMtu( KMTuSizeReceiv );
    obexProtocolPolicy.SetTransmitMtu( KMTuSizeTrans );

    CObexHeader* header = CObexHeader::NewL();
    CleanupStack::PushL( header );
    // _LIT8( KFTPBrowserID, "\xF9\xEC\x7B\xC4\x95\x3C\x11\xd2\x98\x4E\x52\x54\x00\xDC\x9E\x09" )
    header->SetByteSeqL( KBTSUTargetHeader, KFTPBrowserID );

    RArray<CObexHeader*> headerList;
    CleanupClosePushL( headerList );
    headerList.Append( header );

    iConnectObject = CObexNullObject::NewL();
    if ( headerList.Count() > 0 )

```

Remote_Folder_Browsing_over_Bluetooth

```
{
    for ( TInt index = 0; index < headerList.Count(); index++ )
    {
        iConnectObject->AddHeaderL( *headerList[index] );
    }
}

iClient = CObexClient::NewL( info, obexProtocolPolicy );
iClient->SetCallBack( *this );

// Create Connect-object
//
iClient->Connect( *iConnectObject, iStatus ); // Connecting to FTP
SetActive();
iClientState = EHemeBTConnecting;

CleanupStack::Pop(2); // headerList, typeHeader
headerList.Close();
}
```

When we get connection complete indication in our RunL() then we can instruct obex that we are interested to get folder listing. Folder listing request is done in GetObjectL() method.

```
void CHemeBTBrowserObexClient::RunL()
{
    switch ( iClientState )
    {
        case EHemeBTSettingPath:
            GetObjectL();
            break;

        case EHemeBTConnecting:
            {
                GetObjectL();
                iConnectObject->Reset();
                break;
            }

        case EHemeBTGettingFolderList:
            {
                iObserver->GetCompleted( iStatus.Int(), iGetObject );
                break;
            }

        .....
    }
}
```

How to request for folder listing

We need to set the path of the folder which one we are going to browse.

```
/*
 * SetPath()
```

Remote_Folder_Browsing_over_Bluetooth

```
* @param aPathName remote path to be set
*/
void CHemeBTBrowserObexClient::SetPath( const TDesC& aPathName )
{
    if(IsActive())
{
return;
}
    COBex::TSetPathInfo info;
    info.iName.Copy(aPathName);

    info.iNamePresent = ETrue;
    info.iFlags |= 2; // Set "Don't Create" flag as default.

// Parent
if (info.iName.Length() >= 2 && info.iName[0] == '.' && info.iName[1] == '.')
{
    info.iName.Delete(0,2);
    info.iFlags |= 1;
}
// Send object
iClient->SetPath( info, iStatus );
    SetActive
iClientState = EHemeBTSettingPath;
}
```

How to request for folder listing

We need to set right header information for COBexBufObject.

```
/*
* Request for folder listing
*/
void CHemeBTBrowserObexClient::GetObjectL( )
{
    if ( iGetObject )
    {
        iGetObject->Reset();
        delete iGetObject;
        iGetObject = NULL;
    }

    iObjectBuffer = CBufFlat::NewL( KBTSUDataBufferExpandSize );
    iGetObject = COBexBufObject::NewL( iObjectBuffer );

    RArray<COBexHeader*> headerList;
    CleanupClosePushL(headerList);

    COBexHeader* typeHeader = COBexHeader::NewL();
    CleanupStack::PushL( typeHeader );
    //_LIT8( KHemeFolderListing, "x-obex/folder-listing");
    typeHeader->SetByteSeqL( KBTSUTypeHeader, KHemeFolderListing );
    headerList.Append( typeHeader );
    if ( headerList.Count() > 0 )
    {
        for ( TInt index = 0; index < headerList.Count(); index++ )
```

Remote_Folder_Browsing_over_Bluetooth

```
        {
            iGetObject->AddHeaderL( *headerList[index] );
        }
    }
    // Send get request
    //
    iClient->Get( *iGetObject, iStatus );
    SetActive();
    iClientState = EHemeBTGettingFolderList; // Handle in RunL()

    CleanupStack::Pop(2); // headerList, typeHeader
    headerList.Close();
}
```

How to handle folder listing

Remote device sends a XML file contains the list of folders and files in from the target folder of the remote device. We should handle the xml file and display the contents to user. Also we can implement left arrow, right arrow to browse folder to backward and forward direction. The contents of the XML file can be like this according to Obex specification.

```
<folder-listing>
<folder name = ?System? created = ?19961103T141500Z?/>
<file name = ?Jumar.txt? created = ?19971209T090300Z? size = ?6672?/>
<file name = ?Obex.doc? created = ?19970122T102300Z? size = ?41042?/>
</folder-listing>
```

```
/*
 * Do analysis of the xml file
 * @parma aStatus status of the operation
 * @parma aGetResponse CObexBufObject that contains the folder listing
 */
void CHemeBTBrowserFolderContainer::GetCompleted( TInt aStatus, CObexBufObject* aGetResponse )
{
    if(aStatus != KErrNone)
    {
        return;
    }

    _LIT(KAllItemFileName, "templist.dat");
    TFileName tempfile;
    RFs& fsSession = CCoeEnv::Static()->FsSession();
    User::LeaveIfError(fsSession.CreatePrivatePath( EDriveC ) );
    User::LeaveIfError(fsSession.PrivatePath(tempfile));
    tempfile += KAllItemFileName;
    // Create a temp XML folder listing file
    aGetResponse->WriteToFile( tempfile );
    aGetResponse->Reset();

    CBTSUXmlParser* xmlParser = CBTSUXmlParser::NewL();
    CleanupStack::PushL( xmlParser );
    ClearFilesList(); // Clear old file list
    iRemoteFolderList = xmlParser->GetFolderAndFileListL( tempfile );
    fsSession.Delete( tempfile );
}
```

Remote_Folder_Browsing_over_Bluetooth

```
for(TInt i = 0; i <iMessageList->Count(); i++ )
{
    MDesCArray* textArray = iFolderListBox->Model()->ItemTextArray();
    CDesCArray* itemList = static_cast<CDesCArray*>(textArray);
    itemList->Delete( i ); //remove the item
    iFolderListBox->HandleItemRemovalL();
}

TInt capindex = 0;
iMessageList->Reset();
_LIT(KFormat, "%d\t%S\t\t");
TBuf<100> buf(0);
for (capindex=0; capindex < iRemoteFolderList->Count(); capindex++)
{
    buf.Zero();
    buf.Format(KFormat, (*iRemoteFolderList)[capindex].iFolderorFile, (*iRemoteFolderList)[ca
    iMessageList->AppendL(buf);
    iFolderListBox->HandleItemAdditionL();
}
iFolderListBox->DrawNow();
CleanupStack::PopAndDestroy( 1 ); // xmlParser
}
```

Example Applications

There is a complete SDP property/folder browsing application for remote device that can be downloaded from [\[1\]](#).

Here we can find some implementation: [File:HemeBTBrowserObexClient.zip](#)