# Design Document for User Story 5712

## As a User I want the system to use a TBI-CDS Thin-Client so a second logon to the TBI-CDS system is not required once a user is logged into CPRS.

## ASP .NET Pages

N/A

## ASP .NET User Controls

N/A

## ASP .NET Classes

N/A

## C++ Classes

### AppCommandLine.cpp

Code Type: C++ Class

Uses PL/SQL Procedures: N/A

Uses MDWS Calls: N/A

Lines of Code: 36

* CAppCommandLine::CAppCommandLine(void)
  + constructor
* CAppCommandLine::~CAppCommandLine(void)
  + destructor
* void CAppCommandLine::ParseParam( const char\* pszParam,

BOOL bFlag,

BOOL bLast )

* + called for each param passed in on the command line

### IQAESEncrypt.cpp

Code Type: C++ Class

Uses PL/SQL Procedures: N/A

Uses MDWS Calls: N/A

Lines of Code: 430

* CIQAESEncrypt::CIQAESEncrypt(void)
  + construstor
* CIQAESEncrypt::~CIQAESEncrypt(void)
  + destructor
* void CIQAESEncrypt::ReleaseContext(HCRYPTPROV &hProv)
  + releases a context
* bool CIQAESEncrypt::CreateKeyContainer()
  + creates the key container for the application
* BOOL CIQAESEncrypt::CreatePrivateExponentOneKey(LPTSTR szProvider,

DWORD dwProvType,

LPTSTR szContainer,

DWORD dwKeySpec,

HCRYPTPROV \*hProv,

HCRYPTKEY \*hPrivateKey)

The following method was taken directly from Microsoft's support site <http://support.microsoft.com/default.aspx?scid=http://support.microsoft.com:80/support/kb/articles/Q228/7/86.ASP&NoWebContent=1> Sometimes it is convenient to export and import plain text session keys. However, the Microsoft Cryptographic Providers (Base and Enhanced) do not support this feature. Both CryptExportKey() and CryptImportKey() require a valid key handle to encrypt and decrypt the session key, respectively. But by using an "exponent-of-one" private key the same effect can be achieved to encrypt and decrypt the session key.

* bool CIQAESEncrypt::AquireContext(HCRYPTPROV &hProv)
  + aquires a context and creates a key container if necessary
* BOOL CIQAESEncrypt::ImportPlainSessionBlob(HCRYPTPROV hProv,

HCRYPTKEY hPrivateKey,

ALG\_ID dwAlgId,

LPBYTE pbKeyMaterial ,

DWORD dwKeyMaterial ,

HCRYPTKEY \*hSessionKey)

The following method was taken directly from Microsoft's support site <http://support.microsoft.com/default.aspx?scid=http://support.microsoft.com:80/support/kb/articles/Q228/7/86.ASP&NoWebContent=1>

* bool CIQAESEncrypt::GetBytes( LPCTSTR strEncData, //encrypted data

long lFormatType, //1 = ascii, 2 = hex, 3 = utf\_8

PBYTE &pbBuffer,

DWORD &lBufferSize)

* + gets bytes from encrypted data
* bool CIQAESEncrypt::DecryptString(HCRYPTKEY hSessionKey,

LPCTSTR strToDecrypt,

long lFormatType, //1=ascii, 2=hex, 3=utf\_8

CString &strDecrypted)

* + encrypts a string and returns it as hex
* bool CIQAESEncrypt::EncryptString(HCRYPTKEY hSessionKey,

LPCTSTR strStringToEncrypt,

long lFormatType,

CString &strRet)

* + encrypts a string and returns it as hex etc...
* CString CIQAESEncrypt::FormatBytes(PBYTE pbBuffer,

long lpbBufferLen,

long lFormatType)

* + turns bytes into ascii, hex or utf\_8 representation
* void CIQAESEncrypt::DestroyContext(HCRYPTKEY hSessionKey,

HCRYPTKEY hPubPrivKey,

HCRYPTPROV hProv)

* + cleans up context
* bool CIQAESEncrypt::GetContext(CString& strKey,

CString& strIV,

HCRYPTKEY &hSessionKey,

HCRYPTKEY &hPubPrivKey,

HCRYPTPROV &hProv)

* + get context for enc/dec

### MainFrm.cpp

Code Type: C++ Class

Uses PL/SQL Procedures: N/A

Uses MDWS Calls: N/A

Lines of Code: 443

* CMainFrame::CMainFrame()
  + CMainFrame constructor
* CMainFrame::~CMainFrame()
  + CMainFrame destructor
* int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
  + OnCreate, setup menus, toolbars encryption key and load settings
* CString CMainFrame:: GetK()
  + gets the actual key used for encryption
* bool CMainFrame::InitK()
  + init key for enc/dec
* bool CMainFrame::LoadSettings()
  + get settings
* BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
  + pre create window, remove the doc title style

### TBICDSClient.cpp

Code Type: C++ Class

Uses PL/SQL Procedures: N/A

Uses MDWS Calls: N/A

Lines of Code: 155

* BOOL CALLBACK CTBICDSClientApp::FindInstance(HWND hWnd, LPARAM lParam)
  + check for instance already running
* CTBICDSClientApp::CTBICDSClientApp()
  + CTBICDSClientApp construction
* BOOL CTBICDSClientApp::InitInstance()
  + CTBICDSClientApp initialization
* int CTBICDSClientApp::ExitInstance()
  + exit instance

### TBICDSClientDoc.cpp

Code Type: C++ Class

Uses PL/SQL Procedures: N/A

Uses MDWS Calls: N/A

Lines of Code: 54

* CTBICDSClientDoc::CTBICDSClientDoc()
  + CTBICDSClientDoc constructor
* CTBICDSClientDoc::~CTBICDSClientDoc()
  + CTBICDSClientDoc destructor
* BOOL CTBICDSClientDoc::OnNewDocument()
  + on new document handler
* void CTBICDSClientDoc::Serialize(CArchive& ar)
  + CTBICDSClientDoc serialization

### TBICDSClientView.cpp

Code Type: C++ Class

Uses PL/SQL Procedures: N/A

Uses MDWS Calls: N/A

Lines of Code: 121

* CTBICDSClientView::CTBICDSClientView()
  + constructor
* CTBICDSClientView::~CTBICDSClientView()
  + desctructor
* void CTBICDSClientView::BeforeNavigate2(LPDISPATCH pDisp, VARIANT\* URL,

VARIANT\* Flags, VARIANT\* TargetFrameName, VARIANT\* PostData,

VARIANT\* Headers, VARIANT\_BOOL\* Cancel)

* + override beforenavigate2 this will allow us to do processing based on the url we are navigating to
* void CTBICDSClientView::DocumentComplete(LPDISPATCH pDisp, VARIANT\* URL)
  + override document complete this will allow us to do processing after the document is loaded
* BOOL CTBICDSClientView::PreCreateWindow(CREATESTRUCT& cs)
  + precreate window
* void CTBICDSClientView::LogOff()
  + called from the app when the user clicks the X button
* void CTBICDSClientView::CPRSInit()
  + called when the app is activated from the CPRS tools menu
* void CTBICDSClientView::OnInitialUpdate()
  + oninitialupdate
* void CTBICDSClientView::OnRButtonUp(UINT /\* nFlags \*/, CPoint point)
  + onrbuttonup
* void CTBICDSClientView::OnContextMenu(CWnd\* /\* pWnd \*/, CPoint point)
  + on context menu

## Oracle PL/SQL

N/A