Create CSD IAP using CommDB API

Code snippet showing a CSD (circuit switched data) access point creation, using the CommDB API. Note the use of S60FP3 which may be used for second edition feature pack 3 devices.

CsdIap.h

```
#include <commdb.h>
#include <cdbcols.h>
#include <cdbtemp.h>

class CCsdIap : public CBase
{
public:
    static CCsdIap* NewL();
    ~CCsdIap();

    void UpdateTablesL();

private:
    void ConstructL();

    void UpdateModemBearerTableL();
    void UpdateDialOutIspTableL();
    void UpdateNetworkTableL();
    void UpdateIapTableL();
    void UpdateWapAccessPointTableL();
    void UpdateWapIpBearerTableL();

private:
    CCommsDatabase* iCommDb;
    TUint32 iModemId;
    TUint32 iIspId;
    TUint32 iNetworkId;
    TUint32 iIapId;
    TUint32 iWapId;
};
```

CsdIap.cpp

```
#include "CsdIap.h"

#include <e32base.h>

#include <d32comm.h>
#include <faxdefn.h>

#define MODEM_TABLE MODEM_BEARER

_LIT(KIapName, "IapName");
_LIT(KIspDesc, "Iap Description");
_LIT(KTelNum, "+1234567890");
_LIT(KUsername, "user");
_LIT(KPassword, "pass");
_LIT(KIpAddress, "0.0.0.0");
_LIT(KDns1, "0.0.0.0");
_LIT(KDns2, "0.0.0.0");

CCsdIap* CCsdIap::NewL()
{
    CCsdIap* self = new(ELeave) CCsdIap;
    CleanupStack::PushL(self);
    self->ConstructL();
    CleanupStack::Pop(self);

    return self;
}

void CCsdIap::UpdateTablesL()
{
    UpdateModemBearerTableL();
    UpdateDialOutIspTableL();
    UpdateNetworkTableL();
    UpdateIapTableL();
    UpdateWapAccessPointTableL();
    UpdateWapIpBearerTableL();
}

void CCsdIap::ConstructL()
{
    iCommDb = CCommsDatabase::NewL(EDatabaseTypeIAP);
//    commDb->ShowHiddenRecords();
//    User::LeaveIfError(commDb->BeginTransaction());
}

CCsdIap::~CCsdIap()
{
    delete iCommDb;
}

void CCsdIap::UpdateModemBearerTableL()
{
    _LIT(KModemName, "CSD Modem");

    CCommsDbTableView* view = iCommDb->OpenViewMatchingTextLC(TPtrC(MODEM_TABLE), TPtrC(COMMDB_NAME), KModemName);
    TInt error = view->GotoFirstRecord();

    _LIT(KPort, "DATAPORT::0");
    _LIT(KCsy, "DATAPORT");
    _LIT(KTsy, "PHONETSY");

    // If not found, create table
    if (error == KErrNotFound)
    {
        User::LeaveIfError(view->InsertRecord(iModemId));
```


(C) Copyright Nokia 2013. All rights reserved.

```
        view->WriteTextL(TPtrC(COMMDB_NAME), KModemName);

        view->WriteTextL(TPtrC(MODEM_AGENT), _L("csd.agt"));
        view->WriteTextL(TPtrC(MODEM_NIF_NAME), _L("PPP"));
//MODEM_BCA_STACK
        view->WriteTextL(TPtrC(MODEM_PORT_NAME), KPort);
//MODEM_CONTROL_CHANNEL_PORT_NAME
        view->WriteTextL(TPtrC(MODEM_TSY_NAME), KTsy);
        view->WriteTextL(TPtrC(MODEM_CSY_NAME), KCsy);
        view->WriteUintL(TPtrC(MODEM_DATA_BITS), EData8);
        view->WriteUintL(TPtrC(MODEM_STOP_BITS), EStop1);
        view->WriteUintL(TPtrC(MODEM_PARITY), EParityNone);
        view->WriteUintL(TPtrC(MODEM_RATE), EBps19200);
        view->WriteUintL(TPtrC(MODEM_HANDSHAKING), 196);
        view->WriteUintL(TPtrC(MODEM_SPECIAL_RATE), 0);
        view->WriteUintL(TPtrC(MODEM_XON_CHAR), 17);
        view->WriteUintL(TPtrC(MODEM_XOFF_CHAR), 19);
        view->WriteUintL(TPtrC(MODEM_FAX_CLASS_PREF), 0);
        view->WriteUintL(TPtrC(MODEM_SPEAKER_PREF), 0);
        view->WriteUintL(TPtrC(MODEM_SPEAKER_VOL_PREF), 0);
        view->WriteTextL(TPtrC(MODEM_MODEM_INIT_STRING), _L8("ATZ"));
        view->WriteTextL(TPtrC(MODEM_DATA_INIT_STRING), _L8("AT"));
        view->WriteTextL(TPtrC(MODEM_FAX_INIT_STRING), _L8("AT"));
//MODEM_ISP_INIT_STRING
        view->WriteTextL(TPtrC(MODEM_DIAL_PAUSE_LENGTH), _L8("S8="));
//MODEM_CARRIER_TIMEOUT
//MODEM_AUTO_ANSWER_RING_COUNT
        view->WriteTextL(TPtrC(MODEM_SPEAKER_VOL_CONTROL_LOW), _L8("L0"));
        view->WriteTextL(TPtrC(MODEM_SPEAKER_VOL_CONTROL_MEDIUM), _L8("L1"));
        view->WriteTextL(TPtrC(MODEM_SPEAKER_VOL_CONTROL_HIGH), _L8("L2"));
        view->WriteTextL(TPtrC(MODEM_SPEAKER_ALWAYS_OFF), _L8("M0"));
        view->WriteTextL(TPtrC(MODEM_SPEAKER_ON_UNTIL_CARRIER), _L8("M1"));
        view->WriteTextL(TPtrC(MODEM_SPEAKER_ALWAYS_ON), _L8("M2"));
        view->WriteTextL(TPtrC(MODEM_SPEAKER_ON_AFTER_DIAL_UNTIL_CARRIER), _L8("M3"));
        view->WriteTextL(TPtrC(MODEM_DIAL_TONE_WAIT_MODIFIER), _L8("W"));
        view->WriteTextL(TPtrC(MODEM_CALL_PROGRESS_1), _L8("X1"));
        view->WriteTextL(TPtrC(MODEM_CALL_PROGRESS_2), _L8("X2"));
        view->WriteTextL(TPtrC(MODEM_CALL_PROGRESS_3), _L8("X3"));
        view->WriteTextL(TPtrC(MODEM_CALL_PROGRESS_4), _L8("X4"));
        view->WriteTextL(TPtrC(MODEM_ECHO_OFF), _L8("E0"));
        view->WriteTextL(TPtrC(MODEM_VERBOSE_TEXT), _L8("V1"));
        view->WriteTextL(TPtrC(MODEM_QUIET_ON), _L8("Q1"));
        view->WriteTextL(TPtrC(MODEM_QUIET_OFF), _L8("Q0"));
        view->WriteTextL(TPtrC(MODEM_DIAL_COMMAND_STATE_MODIFIER), _L8(";"));
        view->WriteTextL(TPtrC(MODEM_ON_LINE), _L8("O"));
        view->WriteTextL(TPtrC(MODEM_RESET_CONFIGURATION), _L8("Z"));
        view->WriteTextL(TPtrC(MODEM_RETURN_TO_FACTORY_DEFS), _L8("&F"));
        view->WriteTextL(TPtrC(MODEM_DCD_ON_DURING_LINK), _L8("&C1"));
        view->WriteTextL(TPtrC(MODEM_DTR_HANG_UP), _L8("&D2"));
        view->WriteTextL(TPtrC(MODEM_DSR_ALWAYS_ON), _L8("&S0"));
        view->WriteTextL(TPtrC(MODEM_RTS_CTS_HANDSHAKE), _L8("&K3"));
        view->WriteTextL(TPtrC(MODEM_XON_XOFF_HANDSHAKE), _L8("&K4"));
        view->WriteTextL(TPtrC(MODEM_ESCAPE_CHARACTER), _L8("+"));
        view->WriteTextL(TPtrC(MODEM_ESCAPE_GUARD_PERIOD), _L8("S12"));
//MODEM_FAX_CLASS_INTERROGATE
//MODEM_FAX_CLASS
        view->WriteTextL(TPtrC(MODEM_NO_DIAL_TONE), _L8("NO DIAL TONE"));
        view->WriteTextL(TPtrC(MODEM_BUSY), _L8("BUSY"));
        view->WriteTextL(TPtrC(MODEM_NO_ANSWER), _L8("NO ANSWER"));
        view->WriteTextL(TPtrC(MODEM_CARRIER), _L8("CARRIER"));
        view->WriteTextL(TPtrC(MODEM_CONNECT), _L8("CONNECT"));
        view->WriteTextL(TPtrC(MODEM_COMPRESSION_CLASS_5), _L8("COMPRESSION:CLASS 5"));
        view->WriteTextL(TPtrC(MODEM_COMPRESSION_V42BIS), _L8("COMPRESSION:V.42 bis"));
        view->WriteTextL(TPtrC(MODEM_COMPRESSION_NONE), _L8("COMPRESSION:NONE"));
        view->WriteTextL(TPtrC(MODEM_PROTOCOL_LAPD), _L8("PROTOCOL:LAPD"));
        view->WriteTextL(TPtrC(MODEM_PROTOCOL_ALT), _L8("PROTOCOL:ALT"));
        view->WriteTextL(TPtrC(MODEM_PROTOCOL_ALTCELLULAR), _L8("PROTOCOL:ALT-CELLULAR"));
        view->WriteTextL(TPtrC(MODEM_PROTOCOL_NONE), _L8("PROTOCOL:NONE"));
        view->WriteTextL(TPtrC(MODEM_MESSAGE_CENTRE_NUMBER), _L("+34607003110"));
        view->WriteUintL(TPtrC(MODEM_MESSAGE_VALIDITY_PERIOD), 1440);
        view->WriteBoolL(TPtrC(MODEM_MESSAGE_DELIVERY_REPORT), ETrue);
        view->WriteUintL(TPtrC(MODEM_MIN_SIGNAL_LEVEL),0);
        view->WriteUintL(TPtrC(MODEM_SIR_SETTINGS), 0);
        view->WriteUintL(TPtrC(LAST_SOCKET_ACTIVITY_TIMEOUT),300);
        view->WriteUintL(TPtrC(LAST_SESSION_CLOSED_TIMEOUT),1);
        view->WriteUintL(TPtrC(LAST_SOCKET_CLOSED_TIMEOUT),300);
        view->WriteUintL(TPtrC(MODEM_COMM_ROLE), 0);

        User::LeaveIfError(view->PutRecordChanges());
    }
    else
    {
        view->ReadUintL(TPtrC(COMMDB_ID), iModemId);

        User::LeaveIfError(view->UpdateRecord());

        // Put other entries you may want to update
        view->WriteTextL(TPtrC(COMMDB_NAME), KModemName);

        User::LeaveIfError(view->PutRecordChanges());
    }
    CleanupStack::PopAndDestroy(view);
}

void CCsdIap::UpdateDialOutIspTableL()
{
    CCommsDbTableView* view = iCommDb->OpenViewMatchingTextLC(TPtrC(DIAL_OUT_ISP), TPtrC(COMMDB_NAME), KIapName);
    TInt error = view->GotoFirstRecord();

    if (error == KErrNotFound)
    {
        User::LeaveIfError(view->InsertRecord(iIspId));
        view->WriteTextL(TPtrC(COMMDB_NAME), KIapName);
        view->WriteTextL(TPtrC(ISP_DESCRIPTION), KIspDesc);
        view->WriteUintL(TPtrC(ISP_TYPE), EIspTypeInternetAndWAP);
        view->WriteTextL(TPtrC(ISP_DEFAULT_TEL_NUM), KTelNum);
        view->WriteBoolL(TPtrC(ISP_DIAL_RESOLUTION), EFalse);
        view->WriteBoolL(TPtrC(ISP_USE_LOGIN_SCRIPT), EFalse);
//        view->WriteTextL(TPtrC(ISP_LOGIN_SCRIPT), );
        view->WriteBoolL(TPtrC(ISP_PROMPT_FOR_LOGIN), EFalse);
        view->WriteTextL(TPtrC(ISP_LOGIN_NAME), KUsername);
        view->WriteTextL(TPtrC(ISP_LOGIN_PASS), KPassword);
        view->SetNullL(TPtrC(ISP_CHARGECARD));
        view->WriteBoolL(TPtrC(ISP_DISPLAY_PCT),EFalse);

//        view->WriteTextL(TPtrC(ISP_IF_PARAMS), );
        view->WriteTextL(TPtrC(ISP_IF_NETWORKS), _L("ip"));
        view->WriteBoolL(TPtrC(ISP_IF_PROMPT_FOR_AUTH), EFalse);
        view->WriteTextL(TPtrC(ISP_IF_AUTH_NAME), KUsername);
        view->WriteTextL(TPtrC(ISP_IF_AUTH_PASS), KPassword);
```


(C) Copyright Nokia 2013. All rights reserved.

```
        view->WriteUintL(TPtrC(ISP_IF_AUTH_RETRIES), 0);
        view->WriteBoolL(TPtrC(ISP_IF_CALLBACK_ENABLED), EFalse);
        view->WriteUintL(TPtrC(ISP_IF_CALLBACK_TYPE), 1002);
//        view->WriteTextL(TPtrC(ISP_IF_CALLBACK_INFO), );
        view->WriteUintL(TPtrC(ISP_CALLBACK_TIMEOUT), 60000000);
#if defined(S60FP3)
        view->SetNullL(TPtrC(ISP_IF_SERVER_MODE));
#endif
        view->WriteBoolL(TPtrC(ISP_IP_ADDR_FROM_SERVER), ETrue);
        view->WriteTextL(TPtrC(ISP_IP_ADDR), KIpAddress);
//ISP_IP_NETMASK
        view->WriteTextL(TPtrC(ISP_IP_GATEWAY), _L("0.0.0.0"));
        view->WriteBoolL(TPtrC(ISP_IP_DNS_ADDR_FROM_SERVER), ETrue);
        view->WriteTextL(TPtrC(ISP_IP_NAME_SERVER1), KDns1);
        view->WriteTextL(TPtrC(ISP_IP_NAME_SERVER2), KDns2);
        view->SetNullL(TPtrC(ISP_IP6_DNS_ADDR_FROM_SERVER));
//ISP_IP6_NAME_SERVER1
//ISP_IP6_NAME_SERVER2
#if defined(S60FP3)
        view->SetNullL(TPtrC(ISP_IP_ADDR_LEASE_VALID_FROM));
        view->SetNullL(TPtrC(ISP_IP_ADDR_LEASE_VALID_TO));
        view->SetNullL(TPtrC(ISP_CONFIG_DAEMON_MANAGER_NAME));
        view->SetNullL(TPtrC(ISP_CONFIG_DAEMON_NAME));
#endif
        view->WriteBoolL(TPtrC(ISP_ENABLE_IP_HEADER_COMP), EFalse);
        view->WriteBoolL(TPtrC(ISP_ENABLE_LCP_EXTENSIONS), EFalse);
        view->WriteBoolL(TPtrC(ISP_DISABLE_PLAIN_TEXT_AUTH), EFalse);
        view->WriteBoolL(TPtrC(ISP_ENABLE_SW_COMP), EFalse);
        view->WriteUintL(TPtrC(ISP_BEARER_NAME), 0);        // RMobileCall::EProtocolUnspecified
        view->WriteUintL(TPtrC(ISP_BEARER_SPEED), 1);       // RMobileCall::ESpeedAutobauding
        view->WriteUintL(TPtrC(ISP_BEARER_CE), 2);          // RMobileCall::ETchCoding96
//ISP_INIT_STRING
        view->WriteUintL(TPtrC(ISP_BEARER_TYPE), EBearerTypeCSD);
        view->WriteUintL(TPtrC(ISP_CHANNEL_CODING), 0);     // RMobileCall::ETchCodingUnspecified
        view->WriteUintL(TPtrC(ISP_AIUR), 0);               // RMobileCall::EAiurBpsUnspecified
        view->WriteUintL(TPtrC(ISP_REQUESTED_TIME_SLOTS), 0);
        view->WriteUintL(TPtrC(ISP_MAXIMUM_TIME_SLOTS), 0);
//ISP_AUTHENTICATION
        view->WriteUintL(TPtrC(ISP_BEARER_SERVICE), 1);     // RMobileCall::EServiceDataCircuitAsync
        view->WriteUintL(TPtrC(ISP_BEARER_PROTOCOL), 0);    // RMobileCall::EProtocolUnspecified
        view->WriteUintL(TPtrC(ISP_RLP_VERSION), 0);        // RMobileCall::ERLPNotRequested
        view->WriteUintL(TPtrC(ISP_IWF_TO_MS), 0);
        view->WriteUintL(TPtrC(ISP_MS_TO_IWF), 0);
        view->WriteUintL(TPtrC(ISP_ACK_TIMER), 0);
        view->WriteUintL(TPtrC(ISP_RETRANSMISSION_ATTEMPTS), 0);
        view->WriteUintL(TPtrC(ISP_RESEQUENCE_PERIOD), 0);
        view->WriteUintL(TPtrC(ISP_V42_COMPRESSION), 0);    // RMobileCall::EV42bisNeitherDirection
        view->WriteUintL(TPtrC(ISP_V42_CODEWORDS), 0);
        view->WriteUintL(TPtrC(ISP_V42_MAX_LENGTH), 0);
        view->WriteUintL(TPtrC(ISP_ASYMMETRY), 0);          // RMobileCall::EAsymmetryNoPreference
        view->WriteBoolL(TPtrC(ISP_USER_INIT_UPGRADE), EFalse);
        view->WriteBoolL(TPtrC(ISP_USE_EDGE), EFalse);

//        view->SetNullL(TPtrC(SERVICE_ENABLE_LLMNR));

        User::LeaveIfError(view->PutRecordChanges());
    }
    else
    {
        view->ReadUintL(TPtrC(COMMDB_ID), iIspId);

        User::LeaveIfError(view->UpdateRecord());

        // Overwrite existing values (this is useful in case of updating an existing entry)
        view->WriteTextL(TPtrC(COMMDB_NAME), KIapName);
        view->WriteTextL(TPtrC(ISP_DESCRIPTION), KIspDesc);
 view->WriteTextL(TPtrC(ISP_DEFAULT_TEL_NUM), KTelNum);
        view->WriteTextL(TPtrC(ISP_LOGIN_NAME), KUsername);
        view->WriteTextL(TPtrC(ISP_LOGIN_PASS), KPassword);
        view->WriteTextL(TPtrC(ISP_IF_AUTH_NAME), KUsername);
        view->WriteTextL(TPtrC(ISP_IF_AUTH_PASS), KPassword);

 User::LeaveIfError(view->PutRecordChanges());
    }
    CleanupStack::PopAndDestroy(view);
}

void CCsdIap::UpdateNetworkTableL()
{
    CCommsDbTableView* view = iCommDb->OpenViewMatchingTextLC(TPtrC(NETWORK), TPtrC(COMMDB_NAME), KIapName);
    TInt error = view->GotoFirstRecord();

    if (error == KErrNotFound)
    {
        User::LeaveIfError(view->InsertRecord(iNetworkId));
        view->WriteTextL(TPtrC(COMMDB_NAME), KIapName);
        User::LeaveIfError(view->PutRecordChanges());
    }
    else
    {
        view->ReadUintL(TPtrC(COMMDB_ID), iNetworkId);
    }
    CleanupStack::PopAndDestroy(view);
}

void CCsdIap::UpdateIapTableL()
{
    CCommsDbTableView* view = iCommDb->OpenViewMatchingTextLC(TPtrC(IAP), TPtrC(COMMDB_NAME), KIapName);
    TInt error = view->GotoFirstRecord();

    if (error == KErrNotFound)
    {
        User::LeaveIfError(view->InsertRecord(iIapId));
    }
    else
    {
        view->ReadUintL(TPtrC(COMMDB_ID), iIapId);
        User::LeaveIfError(view->UpdateRecord());
    }

    view->WriteTextL(TPtrC(COMMDB_NAME), KIapName);
    view->WriteUintL(TPtrC(IAP_SERVICE), iIspId);
    view->WriteTextL(TPtrC(IAP_SERVICE_TYPE), TPtrC(DIAL_OUT_ISP));
    view->WriteUintL(TPtrC(IAP_BEARER), iModemId);
    view->WriteTextL(TPtrC(IAP_BEARER_TYPE), TPtrC(MODEM_TABLE));
    view->WriteUintL(TPtrC(IAP_NETWORK), iNetworkId);
    view->WriteUintL(TPtrC(IAP_NETWORK_WEIGHTING), 0);

    TUint32 location;
    iCommDb->GetGlobalSettingL(TPtrC(LOCATION_DATA_FAX), location);
```


(C) Copyright Nokia 2013. All rights reserved.

```
    view->WriteUintL(TPtrC(IAP_LOCATION), location);
    User::LeaveIfError(view->PutRecordChanges());

    CleanupStack::PopAndDestroy(view);
}

void CCsdIap::UpdateWapAccessPointTableL()
{
    CCommsDbTableView* view = iCommDb->OpenViewMatchingTextLC(TPtrC(WAP_ACCESS_POINT), TPtrC(COMMDB_NAME), KIapName);
    TInt error = view->GotoFirstRecord();

    if (error == KErrNotFound)
    {
        User::LeaveIfError(view->InsertRecord(iWapId));
        view->WriteTextL(TPtrC(COMMDB_NAME), KIapName);
        view->WriteBoolL(_L("Hidden"), EFalse);
        view->WriteBoolL(_L("ReadOnly"), EFalse);
        view->WriteTextL(TPtrC(WAP_CURRENT_BEARER), TPtrC(WAP_IP_BEARER));
        User::LeaveIfError(view->PutRecordChanges());
    }
    else
    {
        view->ReadUintL(TPtrC(COMMDB_ID), iWapId);
    }
    CleanupStack::PopAndDestroy(view);
}

void CCsdIap::UpdateWapIpBearerTableL()
{
    CCommsDbTableView* view = iCommDb->OpenViewMatchingUintLC(TPtrC(WAP_IP_BEARER), TPtrC(WAP_ACCESS_POINT_ID), iWapId);
    TInt error = view->GotoFirstRecord();

    if (error == KErrNotFound)
    {
        User::LeaveIfError(view->InsertRecord(iWapId));
        view->WriteBoolL(_L("Hidden"), EFalse);
        view->WriteBoolL(_L("ReadOnly"), EFalse);
        view->WriteUintL(TPtrC(WAP_ACCESS_POINT_ID), iWapId);
        view->WriteUintL(TPtrC(WAP_PROXY_PORT), 0);
        view->WriteTextL(TPtrC(WAP_GATEWAY_ADDRESS), _L("0.0.0.0"));
        view->WriteUintL(TPtrC(WAP_IAP), iIapId);
        view->WriteUintL(TPtrC(WAP_WSP_OPTION), 1);
        view->WriteBoolL(TPtrC(WAP_SECURITY), EFalse);
        User::LeaveIfError(view->PutRecordChanges());
    }
    CleanupStack::PopAndDestroy(view);
}
```


(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.