WISE Register Service

Service information

Maemo Platform

```
uint8_t service_uuid_int[] = { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0xab,0xcd };
const char *service_name = "WISE";
const char *service_desc = "Wireless Information Sharing Engine";
const char *service_prov = "Nokia";
```

S60 Platform

```
TUint32 KServiceID(0xabcd);
_LIT(KServiceName, "WISE");
_LIT(KServiceDesc, "Wireless Information Sharing Engine");
_LIT(KServiceProv, "Nokia");
```

Variables needed

```
/* Maemo */                                  /* S60 */
uint8_t rfcomm_channel TInt rfcommChannel

uuid_t root_uuid,
       l2cap_uuid,
       rfcomm_uuid,
       svc_uuid;                              TUUID serviceUUID;

sdp_list_t *l2cap_list = NULL,
           *rfcomm_list = NULL,
           *root_list = NULL,
           *proto_list = NULL,                CSdpAttrValueDES* protocolDescriptorList;
           *access_proto_list = NULL;

sdp_data_t *channel = 0;                      TBuf8<1> channel;

sdp_session_t *session = 0;                   RSdp sdp;
sdp_record_t *record = NULL;                  TSdpServRecordHandle record;
                                              RSdpDatabase sdpDB;
                                              TInt recordState;
```

Connect to SDP server

Variables used

```
/* Maemo */                                  /* S60 */
session sdp
                                              sdpDB
```

Maemo Platform

```
/* connect to the local SDP server */
session = sdp_connect(BDADDR_ANY,BDADDR_LOCAL,SDP_RETRY_IF_BUSY);
```

S60 Platform

```
// open sdp session
User::LeaveIfError(sdp.Connect());

// open sdp database session
User::LeaveIfError(sdpDB.Open(sdp));
```

Set service UUID

Variables used

```
/* Maemo */                                  /* S60 */
service_uuid_int KServiceID
svc_uuid serviceUUID

record record
                                              sdpDB
```

Maemo Platform

(C) Copyright Nokia 2013. All rights reserved.

```
/* create a record of the correct service class */
record = sdp_record_alloc();
sdp_uuid128_create(&svc_uuid,&service_uuid_int);
sdp_set_service_id(record,svc_uuid);
```

S60 Platform

```
// create a record of the correct service class
TUUID serviceUUID(0x0000,0x0000,0x0000,KServiceID);
sdpDB.CreateServiceRecordL(serviceUUID,record);
```

Create protocol list for service

Variables used

```
/* Maemo */                                      /* S60 */
rfcomm_channel rfcommChannel
channel channel
access_proto_list
proto_list protocolDescriptorList
l2cap_list
rfcomm_list
root_uuid
l2cap_uuid
rfcomm_uuid
```

Maemo Platform

```
/* set l2cap information */
sdp_uuid16_create(&l2cap_uuid,L2CAP_UUID);
l2cap_list = sdp_list_append(0,&l2cap_uuid);
proto_list = sdp_list_append(0,l2cap_list);

/* set rfcomm information */
sdp_uuid16_create(&rfcomm_uuid,RFCOMM_UUID);
channel = sdp_data_alloc(SDP_UINT8,&rfcomm_channel);
rfcomm_list = sdp_list_append(0,&rfcomm_uuid);
sdp_list_append(rfcomm_list,channel);
sdp_list_append(proto_list,rfcomm_list);
```

S60 Platform

```
// create protocol list for our service
protocolDescriptorList = CSdpAttrValueDES::NewDESL(NULL);
CleanupStack::PushL(protocolDescriptorList);

channel.Append((TChar)rfcommChannel);
protocolDescriptorList
  ->StartListL()   //  list of protocols required for this method
    ->BuildDESL()
    ->StartListL()
      ->BuildUUIDL(KL2CAP)
    ->EndListL()

    ->BuildDESL()
    ->StartListL()
      ->BuildUUIDL(KRFCOMM)
      ->BuildUintL(channel)
    ->EndListL()
  ->EndListL();
```

Set protocol information to the service record

Variables used

```
/* Maemo */                                  /* S60 */
channel
l2cap_list
rfcomm_list
root_list
access_proto_list protocolDescriptorList
proto_list
record record
                                             sdpDB
```

Maemo Platform

```
/* set protocol information to the service record */
access_proto_list = sdp_list_append(0,proto_list);
sdp_set_access_protos(record,access_proto_list);

/* clean up */
sdp_data_free(channel);
sdp_list_free(l2cap_list,0);
sdp_list_free(rfcomm_list,0);
sdp_list_free(root_list,0);
sdp_list_free(access_proto_list,0);
```

S60 Platform

```
// set protocol information to the service record
sdpDB.UpdateAttributeL(record,KSdpAttrIdProtocolDescriptorList,
    *protocolDescriptorList);
```


(C) Copyright Nokia 2013. All rights reserved.

```
/* clean up */
CleanupStack::PopAndDestroy(protocolDescriptorList);
```

Set service information

Variables used

```
/* Maemo */                                  /* S60 */
service_name KServiceName
service_dsc KServiceDesc
service_prov KServiceProv

record record
                                            sdpDB
```

Maemo Platform

```
/* set the name, provider, and description */
sdp_set_info_attr(record,service_name,service_prov,service_desc);
```

S60 Platform

```
// add a name to the record
sdpDB.UpdateAttributeL(record,
    KSdpAttrIdBasePrimaryLanguage +
    KSdpAttrIdOffsetServiceName,
    KServiceName);

// add a description to the record
sdpDB.UpdateAttributeL(record,
    KSdpAttrIdBasePrimaryLanguage +
    KSdpAttrIdOffsetServiceDescription,
    KServiceDesc);

// add a provider to the record
sdpDB.UpdateAttributeL(record,
    KSdpAttrIdBasePrimaryLanguage +
    KSdpAttrIdOffsetProviderName,
    KServiceProv);
```

Make service public

Variables used

```
/* Maemo */                                  /* S60 */
root_uuid
root_list
                                            recordState
record record
                                            sdpDB
```

Maemo Platform

```
/* make the service record publicly browsable */
sdp_uuid16_create(&root_uuid,PUBLIC_BROWSE_GROUP);
root_list = sdp_list_append(0,&root_uuid);
sdp_set_browse_groups(record,root_list);
```

S60 Platform

```
// set availability
sdpDB.UpdateAttributeL(record,KSdpAttrIdServiceAvailability,0xFF);
// mark record changed
sdpDB.UpdateAttributeL(record,KSdpAttrIdServiceRecordState,recordState);
```

Register service

```
Variables used
```

```
/* Maemo */
session
record
```

Maemo Platform

```
sdp_record_register(session,record,0);
```

S60 Platform

(C) Copyright Nokia 2013. All rights reserved.

Not needed separately

Service already registered when service record is created (*Set service UUID*).

```
sdpDB.CreateServiceRecordL(serviceUUID,record);
```

Complete WISE functions

Maemo Platform

```
typedef struct
{
    int registered;           /* Server registered to the SDP server */
    int server_connected;     /* Server connected to a client */
    int client_connected;     /* Server socket active */
    sdp_session_t* session;   /* SDP session */
    sdp_record_t* record;     /* SDP record */
    int server_socket;        /* Server_socket */
    int client_socket;        /* Client socket */
    char client_address[18];  /* Client Bluetooth address */
} wise_server;
```

```
/**
* Register WISE server to the SDP server
*/

int server_register(wise_server* server,uint8_t rfcomm_channel)
{
    uint8_t service_uuid_int[] = { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0xab,0xcd };
    const char *service_name = "WISE";
    const char *service_dsc = "Wireless Information Sharing Engine";
    const char *service_prov = "Nokia";

    uuid_t root_uuid,l2cap_uuid,rfcomm_uuid,svc_uuid;

    sdp_list_t *l2cap_list=0,
               *rfcomm_list=0,
               *root_list=0,
               *proto_list=0,
               *access_proto_list=0;

    sdp_data_t *channel=0;
    int error=WISE_OK;
    sdp_session_t *session=0;

    sdp_record_t *record = sdp_record_alloc();
    if ( record==NULL ) return WISE_CONNECTION_ERROR;

    /* set the general service ID */
    sdp_uuid128_create(&svc_uuid,&service_uuid_int);
    sdp_set_service_id(record,svc_uuid);

    /* make the service record publicly browsable */
    sdp_uuid16_create(&root_uuid,PUBLIC_BROWSE_GROUP);
    root_list = sdp_list_append(0,&root_uuid);
    sdp_set_browse_groups(record,root_list);

    /* set l2cap information */
    sdp_uuid16_create(&l2cap_uuid,L2CAP_UUID);
    l2cap_list = sdp_list_append(0,&l2cap_uuid);
    proto_list = sdp_list_append(0,l2cap_list);

    /* set rfcomm information */
    sdp_uuid16_create(&rfcomm_uuid,RFCOMM_UUID);
    channel = sdp_data_alloc(SDP_UINT8,&rfcomm_channel);
    rfcomm_list = sdp_list_append(0,&rfcomm_uuid);
    sdp_list_append(rfcomm_list,channel);
    sdp_list_append(proto_list,rfcomm_list);

    /* attach protocol information to service record */
    access_proto_list = sdp_list_append(0,proto_list);
    sdp_set_access_protos(record,access_proto_list);

    /* set the name, provider, and description */
    sdp_set_info_attr(record,service_name,service_prov,service_dsc);

    /* connect to the local SDP server, register the service record */
    session = sdp_connect(BDADDR_ANY,BDADDR_LOCAL,SDP_RETRY_IF_BUSY);

    if ( session==NULL ) error = WISE_CONNECTION_ERROR;
    else error = sdp_record_register(session,record,0);

    /* cleanup */
    sdp_data_free(channel);
    sdp_list_free(l2cap_list,0);
    sdp_list_free(rfcomm_list,0);
    sdp_list_free(root_list,0);
    sdp_list_free(access_proto_list,0);

    if ( error==WISE_OK && session!=NULL )
    {
        server->session = session;
        server->record = record;
        server->registered = TRUE;
    }
    return error;
}
```

Links

- Archived:Wireless Information Sharing Engine
- WISE Protocol

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.