```
public class DeviceConnectActivity extends CommonEventBaseFragmentActivity implements
ViewPager.OnPageChangeListener
{
   public static boolean gDeviceConnect = false;
   private DeviceConnectFragment deviceConnectFragment = null;
   private BluetoothNetworkFragment bluetoothNetworkFragment = null;
   private CommonFragmentAdapter mFragmentAdapter;
   @Override
   protected void onCreate(Bundle savedInstanceState)
   {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_device_connect);
      gDeviceConnect = true;
      initialUI();
      if (!ConfigDeviceManage.GetInstance().getDeviceModelType().isSupportNfc()) {
         CommonBaseActivity.mDeviceNFC.init(this);
         CommonBaseActivity.mDeviceNFC.setNfcCallback(this);
      }
   }
   private void initialUI()
   {
      mFragmentAdapter = new CommonFragmentAdapter(getSupportFragmentManager());
      deviceConnectFragment = new DeviceConnectFragment();
      mFragmentAdapter.addFragment(deviceConnectFragment);
      if (AppFinalUtil.getAppID().isMiniApp())
      {
         bluetoothNetworkFragment = new BluetoothNetworkFragment();
         mFragmentAdapter.addFragment(bluetoothNetworkFragment);
      }
      ViewPager viewPager = findViewById(R.id.viewPager_Fragment);
      viewPager.setAdapter(mFragmentAdapter);
      TabLayout tabLayout = findViewById(R.id.tabLayout_Fragment);
      tabLayout.setupWithViewPager(viewPager);
      if (mFragmentAdapter.getCount() <= 1)
      {
         tabLayout.setVisibility(View.GONE);
      }
      else {
         if                (DeviceManage.GetInstance().getConnectedStatus()                ==
ConnectListener.CommanderStatus.SUCCESS)
         {
            viewPager.setCurrentItem(1);
         }
      }
      CustomWaittingLayout waittingLayout = findViewById(R.id.waittingLayout);
      waittingLayout.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            DeviceManage.GetInstance().disConnect();
            showWaitingDialog(false);
```

```
            }
        });
        CustomCommandWaittingLayout commandWaittingLayout =
findViewById(R.id.commandWaittingLayout);
        commandWaittingLayout.setOnCommandListener(new
CustomCommandWaittingLayout.OnCommandListener() {
            @Override
            public void onCommandStop() {
                deviceConnectFragment.onButtonStop();
            }
            @Override
            public void onCommandFinish(boolean finish) {
                if (finish) {
                    finish();
                }
            }
        });
    }
    protected void showWaitingDialog(boolean isShow)
    {
        setVisibility(R.id.waittingLayout, isShow ?View.VISIBLE : View.GONE);
    }
    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        ViewPager viewPager = findViewById(R.id.viewPager_Fragment);
        mFragmentAdapter.getItem(viewPager.getCurrentItem()).onActivityResult((requestCode &
0xFFFF), resultCode, data);
    }
    @Override
    public void onDeviceFound(String name, String address) {
        DeviceModelType type = ConfigDeviceManage.GetInstance().getDeviceModelType();
        if (deviceConnectFragment != null && type.isSupportNfc() &&
CommandSendManage.getInstance().size() <= 0)
        {
            if (DeviceManage.GetInstance().isConnected())
            {
                if (DeviceManage.GetInstance().getConnectedStatus() !=
ConnectListener.CommanderStatus.FAIL &&
                        address.equals(ConfigDeviceManage.GetInstance().getBluetoothAddress()))
                    return;
                deviceConnectFragment.onButtonStop();
            }
            ConfigDeviceManage.GetInstance().setBluetoothName(name);
            ConfigDeviceManage.GetInstance().setBluetoothAddress(address);
            deviceConnectFragment.updateDeviceLinker(DeviceLinkerType.BLUETOOTH);
            deviceConnectFragment.onButtonConnect();
        }
    }

```

```
    public void onEventMainThread(DeviceUpdateEvent.UpdateNetworkConnectStatus obj)
    {
        if (obj == null || bluetoothNetworkFragment == null)
            return;
        Toast toast = Toast.makeText(this, obj.getStatus(), Toast.LENGTH_SHORT);
        toast.setGravity(Gravity.CENTER, 0, 0);
        toast.show();
        bluetoothNetworkFragment.updateConnectUI();
    }
    public void onEventMainThread(DeviceUpdateEvent.ReceiverNetworkData obj)
    {
        if (obj == null || bluetoothNetworkFragment == null)
            return;
        bluetoothNetworkFragment.receiverNetworkData(obj.getSize());
    }
    public void onEventMainThread(DeviceUpdateEvent.UpdateMountPointSuccessStatus obj) {
        if (obj == null || bluetoothNetworkFragment == null)
            return;
        if (obj.IsGetMountPointSuccess()) {
            SourecTableManage.getInstance().SaveSourceTable();
            makeTextString(R.string.string_prompt_get_mount_point_list_succeeded,
Gravity.CENTER);
        } else {
            makeTextString(R.string.string_prompt_get_mount_point_failed, Gravity.CENTER);
        }
        showWaitingDialog(false);
        bluetoothNetworkFragment.updateMountPointFinish();
    }
    public void onEventMainThread(DeviceUpdateEvent.DeviceConnectResultStatus eventStatus) {
        showWaitingDialog(false);
        if (eventStatus.getIsConnectSuccess())
        {
            DeviceManage.GetInstance().setConnectState(true);
            if (deviceConnectFragment != null)
                deviceConnectFragment.updateButtonState();
            CustomCommandWaittingLayout                commandWaittingLayout                =
findViewById(R.id.commandWaittingLayout);
            commandWaittingLayout.startSendCommand(bluetoothNetworkFragment == null, true);
        } else {
            if (DeviceManage.GetInstance().getCurLinkerType() == DeviceLinkerType.LOCAL) {
                makeTextString(getString(R.string.toast_internal_gps_not_enabled));
            } else {
                makeTextString(getString(R.string.string_prompt_connection_failed));
            }
        }
    }
    @Override
    public void finish()
    {
        if (deviceConnectFragment != null && deviceConnectFragment.cancelBatchSelectMode())
```

```
            return;
        CustomWaittingLayout waittingLayout = findViewById(R.id.waittingLayout);
        if (waittingLayout.getVisibility() == View.VISIBLE)
            return;
        CustomCommandWaittingLayout commandWaittingLayout =
findViewById(R.id.commandWaittingLayout);
        if (commandWaittingLayout.getVisibility() == View.VISIBLE)
            return;
        commandWaittingLayout.setOnCommandListener(null);
        ConfigDeviceManage.GetInstance().saveConfig();
        gDeviceConnect = false;
        super.finish();
    }
    @Override
    public void onPageScrolled(int position, float positionOffset, int positionOffsetPixels) {
        if (bluetoothNetworkFragment != null) {
            ViewPager viewPager = findViewById(R.id.viewPager_Fragment);
            CommonV4Fragment commonV4Fragment =
mFragmentAdapter.getItem(viewPager.getCurrentItem());
            CustomWaittingLayout waittingLayout = findViewById(R.id.waittingLayout);
            if (commonV4Fragment == deviceConnectFragment)
            {
                waittingLayout.setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        DeviceManage.GetInstance().disConnect();
                        showWaitingDialog(false);
                    }
                });
            }
            else {
                waittingLayout.setOnClickListener(null);
            }
        }
    }
    @Override
    public void onPageSelected(int position) {
    }
    @Override
    public void onPageScrollStateChanged(int state) {
    }
}
public class BluetoothSearchActivity extends CommonGridBaseActivity
        implements View.OnClickListener {
     private DeviceModelType deviceModelType = DeviceModelType.TYPE_UNKNOWN;
     private BroadcastReceiver mBluetoothReceiver = null;
     private ArrayList<tagDeviceNode> deviceNodeArrayList = new ArrayList<>();
     @Override
     protected int getContentView() {
         return R.layout.activity_bluetooth_search;
```

```
    }
    @Override
    protected void initCustom() {
        deviceModelType =
DeviceModelType.swigToEnum(getIntent().getIntExtra("DeviceModelType",
DeviceModelType.TYPE_UNKNOWN.swigValue()));
        try {
            if (mGridItemListAdapter == null) {
                mGridItemListAdapter = new CustomGridBluetoothItemAdapter(this, this,
deviceNodeArrayList);
            }
        } catch (Exception e) {
            e.printStackTrace();
            return;
        }
        mGridView.setAdapter(mGridItemListAdapter);
        if (mBluetoothReceiver == null) {
            IntentFilter intentFilter = new IntentFilter();
            intentFilter.addAction(BluetoothDevice.ACTION_FOUND);
            intentFilter.addAction(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
            intentFilter.addAction(BluetoothDevice.ACTION_BOND_STATE_CHANGED);
            intentFilter.addAction(BluetoothAdapter.ACTION_SCAN_MODE_CHANGED);
            intentFilter.addAction(BluetoothAdapter.ACTION_STATE_CHANGED);
            mBluetoothReceiver = new BluetoothStateReceiver();
            registerReceiver(mBluetoothReceiver, intentFilter);
        }
        updatePairedDeviceList();
        onButtonSearchBluetooth();
    }
    @Override
    public void onClickItem(int position) {
        if (mGridItemListAdapter.isBatchSelectMode()) {
            mGridItemListAdapter.onButtonCheckItem(position);
        } else {
            int selectPosition = position;
            if (mGridItemListAdapter.getSelectedPosition() >= 0/* == position*/) {
                selectPosition = -1;
            }
            onClickApply(selectPosition);
        }
    }
    private void updatePairedDeviceList() {
        deviceNodeArrayList.clear();
        for (int i = 0; deviceNodeArrayList.size() < 5 && i <
ConfigDeviceManage.GetInstance().getBluetoothDeviceCount(); i++) {
            tagDeviceNode deviceNode = ConfigDeviceManage.GetInstance().getBluetoothDevice(i);
            deviceNodeArrayList.add(deviceNode);
        }
        mGridItemListAdapter.setSelectedPosition(-1);
        if (AppFinalUtil.getAppID().isXSurvey()) {
```

```
            setVisibility(R.id.linearLayout_BluetoothList, deviceNodeArrayList.size() > 0 ?
View.VISIBLE : View.GONE);
        } else {
            setVisibility(R.id.linearLayout_BluetoothList, View.GONE);
        }
    }
    @Override
    public void finish() {
        if (mBluetoothReceiver != null) {
            unregisterReceiver(mBluetoothReceiver);
            mBluetoothReceiver = null;
        }
        BluetoothAdapter adapter = BluetoothAdapter.getDefaultAdapter();
        if (adapter != null && adapter.isDiscovering()) {
            adapter.cancelDiscovery();
        }
        super.finish();
    }
    @Override
    protected void onClickApply(int selectIndex) {
        if (selectIndex < 0)
            return;
        tagDeviceNode deviceNode = (tagDeviceNode)mGridItemListAdapter.getItem(selectIndex);
        finish(deviceNode.strName, deviceNode.strAddress);
    }
    @Override
    protected void onButtonSelect() {
    }
    @Override
    protected void onButtonAdd() {
    }
    @Override
    protected void onButtonDelete(int index) {
    }
    @Override
    protected void onButtonDelete(ArrayList<Integer> indexs) {
        if (indexs.size() >= 5)
        {
            ConfigDeviceManage.GetInstance().removeAll();
        }
        else {
            for (int i = indexs.size() - 1; i >= 0; i--) {
                tagDeviceNode deviceNode = (tagDeviceNode)
mGridItemListAdapter.getItem(indexs.get(i));
ConfigDeviceManage.GetInstance().removeBluetoothDevice(deviceNode.strAddress);
            }
        }
        updatePairedDeviceList();
    }
    @Override
```

```
    protected void onButtonShare() {
    }
    private static void unpairDevice(BluetoothDevice device) {
        try {
            Method createBondMethod = device.getClass().getMethod("cancelBondProcess");
            createBondMethod.invoke(device);
        } catch (Exception e) {
        }
    }
    @Override
    protected void onButtonImport() {
    }
    @Override
    protected void onButtonExport() {
    }
    @Override
    protected void onButtonOK() {
        onButtonSearchBluetooth();
    }
    private void onButtonSearchBluetooth() {
        BluetoothAdapter adapter = BluetoothAdapter.getDefaultAdapter();
        if (adapter == null) {
            return;
        }
        if (adapter.isDiscovering()) {
            adapter.cancelDiscovery();
            setTextViewText(R.id.button_OK, getString(R.string.button_refresh));
            return;
        }
        if (!checkGpsPermission()) {
            makeTextString(R.string.toast_gps_permission_check);
            return;
        }
        if (!checkOpen()) {
            makeTextString(R.string.toast_turning_on_bluetooth);
            return;
        }
        adapter.startDiscovery();
        setTextViewText(R.id.button_OK, getString(R.string.string_prompt_searching));
        CustomTextViewListLayout                    viewListLayoutAvailable                    =
findViewById(R.id.viewListLayoutAvailable);
        viewListLayoutAvailable.clear();
    }
    private boolean checkGpsPermission(){
        if(android.os.Build.VERSION.SDK_INT >= 23) {
            if(ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION)!= PackageManager.PERMISSION_GRANTED){
                ActivityCompat.requestPermissions(this,new
String[]{Manifest.permission.ACCESS_FINE_LOCATION},101);
                return false;
```

```
            }
            return true;
        }
        return true;
    }
    private boolean mBeWaitingWifiOpen = false;
    private boolean checkOpen() {
        if (mBeWaitingWifiOpen)
            return false;
        BluetoothAdapter mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
        if (mBluetoothAdapter == null) {
            return false;
        }
        if (!mBluetoothAdapter.isEnabled()) {
            mBluetoothAdapter.enable();
            mBeWaitingWifiOpen = true;
            return false;
        }
        return true;
    }
    private void finish(String strName, String strAddress)
    {
        Intent intent = new Intent();
        intent.putExtra("BluetoothName", strName);
        intent.putExtra("BluetoothAddress", strAddress);
        setResult(AppFinalUtil.RETURN_CODE_VALUE_BACK, intent);
        finish();
    }
    @Override
    public void onClickEdit() {
    }
    @Override
    public void onClickShare() {
    }
    public class BluetoothStateReceiver extends BroadcastReceiver {
        ArrayList<String> bluetoothDeviceList = new ArrayList<>();
        @Override
        public void onReceive(Context context, Intent intent) {
            switch (intent.getAction()) {
                case BluetoothDevice.ACTION_FOUND:
                    BluetoothDevice                          device                          =
intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
                    tagDeviceNode deviceNode = new tagDeviceNode();
                    deviceNode.strName = device.getName();
                    if (deviceNode.strName == null || deviceNode.strName.equals("")) {
                        deviceNode.strName = "Unknow Device";
                        return;
                    }
                    if (deviceModelType == DeviceModelType.TYPE_MODEL_ZX_RTK)
                    {
```

```
                        if (deviceNode.strName.indexOf("Z") != 0)
                            return;
                    }
                    deviceNode.strAddress = device.getAddress();
                    LayoutInflater inflater = LayoutInflater.from(context);
                    View                                convertView                                =
inflater.inflate(R.layout.layout_gridview_bluetooth_item,null);
                    TextView textView = convertView.findViewById(R.id.textView_Name);
                    textView.setText(deviceNode.strName);
                    textView = convertView.findViewById(R.id.textView_Address);
                    textView.setText(deviceNode.strAddress);
                    convertView.setOnClickListener(new View.OnClickListener() {
                        @Override
                        public void onClick(View v) {
                            TextView textView = v.findViewById(R.id.textView_Name);
                            String strName = textView.getText().toString();
                            textView = v.findViewById(R.id.textView_Address);
                            String strAddress = textView.getText().toString();
                            finish(strName, strAddress);
                        }
                    });
                    CustomTextViewListLayout            viewListLayoutAvailable            =
findViewById(R.id.viewListLayoutAvailable);
                    View viewLine = convertView.findViewById(R.id.viewLine);
                    if (viewListLayoutAvailable.getCount() <= 0)
                    {
                        viewLine.setVisibility(View.GONE);
                        bluetoothDeviceList.clear();
                    }else {
                        viewLine.setVisibility(View.VISIBLE);
                    }
                    for (int i=0; i<bluetoothDeviceList.size(); i++)
                    {
                        if
(deviceNode.strAddress.equalsIgnoreCase(bluetoothDeviceList.get(i)))
                        {
                            return;
                        }
                    }
                    bluetoothDeviceList.add(deviceNode.strAddress);
                    viewListLayoutAvailable.add(convertView);
                    break;
                case BluetoothAdapter.ACTION_DISCOVERY_FINISHED:
                    setTextViewText(R.id.button_OK, getString(R.string.button_refresh));
                    break;
                case BluetoothAdapter.ACTION_STATE_CHANGED:
                    int blueState = intent.getIntExtra(BluetoothAdapter.EXTRA_STATE, 0);
                    switch(blueState){
                        case BluetoothAdapter.STATE_TURNING_ON:
                            break;
```

```
                        case BluetoothAdapter.STATE_ON:
                            if (mBeWaitingWifiOpen) {
                                mBeWaitingWifiOpen = false;
                                onButtonSearchBluetooth();
                            }
                            break;
                        case BluetoothAdapter.STATE_TURNING_OFF:
                            break;
                        case BluetoothAdapter.STATE_OFF:
                            break;
                    }
                    break;
                }
            }
        }
    }
}
public class BluetoothNetworkFragment extends CommonV4Fragment implements OnClickListener
{
    public tagDataLinkParameter dataLinkParameter = new tagDataLinkParameter();
    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, Bundle
savedInstanceState) {
        if (mInfoView != null)
            return mInfoView;
        mInfoView = inflater.inflate(R.layout.layout_bluetooth_network, container,false);
dataLinkParameter.set(DeviceCommandParse.getInstance().mConfigWorkMode.dataLinkParameter);
        DeviceManage.GetInstance().setPageInBlueToothSetting(true);
        dataLinkParameter.bluetoothParam.set(ConfigNtripParameter.getInstance());
        initializeUI();
        updateValueToUI();
        return mInfoView;
    }
    @Override
    public String getTitle() {
        return AppFinalUtil.getString(R.string.string_differential_data);
    }
    @Override
    public void updateValueToUI()
    {
        if (mInfoView == null)
            return;
        CustomTextViewLayoutSelect                    layoutSelectDataLink                    =
mInfoView.findViewById(R.id.layoutSelect_DataLink);
        layoutSelectDataLink.setSelectedId(dataLinkParameter.Type.swigValue());
        updateConnectUI();
    }
    public void updateValueFormUI()
    {
        if (dataLinkParameter.Type == EDataLinkType.ExtendSource)
```

```
        {
ConfigNtripParameter.getInstance().setAutoConnect(getCheckButtonChecked(R.id.checkButton_Auto));
            dataLinkParameter.bluetoothParam.strMountPoint = getTextViewString(R.id.layoutSelectEdit_MountPoint);
            ConfigNtripParameter.getInstance().set(dataLinkParameter.bluetoothParam);
            ConfigNtripParameter.getInstance().saveConfig();
        }
    }
    private void initializeUI()
    {
        initClickListener(R.id.button_MountPoint,this);
        initClickListener(R.id.button_Start,this);
        initClickListener(R.id.button_Stop,this);
        setCheckButtonChecked(R.id.checkButton_Auto,
ConfigNtripParameter.getInstance().isAutoConnect());
        CustomTimerView timerView_Rev = mInfoView.findViewById(R.id.timerView_Rev);
        timerView_Rev.setMode(99);
        timerView_Rev.setMaxValue(8000);
        initializeMountPointList();
        CustomTextViewLayoutSelect layoutSelectDataLink = mInfoView.findViewById(R.id.layoutSelect_DataLink);
        layoutSelectDataLink.registerItemSelectedInterface(new
CustomTextViewLayoutSelect.OnItemSelectedCallback() {
            @Override
            public void onItemSelected(View view, String label, int id) {
                dataLinkParameter.Type = EDataLinkType.swigToEnum(id);
                CustomTextViewListLayout linearLayoutDataLink = mInfoView.findViewById(R.id.linearLayout_DataLink);
                linearLayoutDataLink.clear();
                boolean bDisplay = false;
                boolean bNtripMountPoint = false;
                switch (dataLinkParameter.Type) {
                    case ExtendSource: {
                        bDisplay = true;
                        linearLayoutDataLink.add(1, StringFunction.format("%s:%s",
getString(R.string.string_connect_mode),
dataLinkParameter.bluetoothParam.connectMode.getTitle()), "", "", "");
                        switch (dataLinkParameter.bluetoothParam.connectMode) {
                            case CORS_CONNECT_MODE_NTRIP: {
                                bNtripMountPoint = true;
                                setTextViewText(R.id.layoutSelectEdit_MountPoint,
dataLinkParameter.bluetoothParam.strMountPoint);
                                linearLayoutDataLink.add(2, StringFunction.format("%s:%s",
getString(R.string.string_server_ip), dataLinkParameter.bluetoothParam.strIP),
StringFunction.format("%s:%d", getString(R.string.string_remote_port),
dataLinkParameter.bluetoothParam.nPort), "", "");
                                linearLayoutDataLink.add(2, StringFunction.format("%s:%s",
getString(R.string.string_user), dataLinkParameter.bluetoothParam.strUser),
StringFunction.format("%s:%s", getString(R.string.string_password), "******"), "", "");
```

```
                    }
                    break;
                    case CORS_CONNECT_MODE_TCP: {
                        linearLayoutDataLink.add(2,          StringFunction.format("%s:%s",
getString(R.string.string_server_ip),              dataLinkParameter.bluetoothParam.strIP),
StringFunction.format("%s:%d",                      getString(R.string.string_remote_port),
dataLinkParameter.bluetoothParam.nPort), "", "");
                    }
                    break;
                    case CORS_CONNECT_MODE_ZHD://中海达
                    {
                        linearLayoutDataLink.add(2,          StringFunction.format("%s:%s",
getString(R.string.string_server_ip),              dataLinkParameter.bluetoothParam.strIP),
StringFunction.format("%s:%d",                      getString(R.string.string_remote_port),
dataLinkParameter.bluetoothParam.nPort), "", "");
                        linearLayoutDataLink.add(2,          StringFunction.format("%s:%s",
getString(R.string.string_group_number),          dataLinkParameter.bluetoothParam.strUser),
StringFunction.format("%s:%s",                   getString(R.string.string_sub_group_number),
dataLinkParameter.bluetoothParam.strPassword), "", "");
                    }
                    break;
                    case CORS_CONNECT_MODE_HUACE: {
                        linearLayoutDataLink.add(2,          StringFunction.format("%s:%s",
getString(R.string.string_server_ip),              dataLinkParameter.bluetoothParam.strIP),
StringFunction.format("%s:%d",                      getString(R.string.string_remote_port),
dataLinkParameter.bluetoothParam.nPort), "", "");
                        linearLayoutDataLink.add(1,          StringFunction.format("%s:%s",
getString(R.string.string_huace_base_id),   dataLinkParameter.bluetoothParam.strMountPoint),
"", "", "");
                    }
                    break;
                }
            }
            break;
            default:
                break;
        }
        linearLayoutDataLink.setVisibility(bDisplay?View.VISIBLE: View.GONE);
        setVisibility(R.id.linearLayout_MountPoint,
bNtripMountPoint?View.VISIBLE:View.GONE);
        setVisibility(R.id.linearLayout_Cors,          dataLinkParameter.Type          ==
EDataLinkType.ExtendSource?View.VISIBLE:View.GONE);
      }
    });
layoutSelectDataLink.addItem(getString(R.string.title_bluetooth_datalink),EDataLinkType.Ext
endSource.swigValue());
    CustomTextViewListLayout                    linearLayoutDataLink                    =
mInfoView.findViewById(R.id.linearLayout_DataLink);
    linearLayoutDataLink.setOnClickListener(new OnClickListener() {
      @Override
```

```
        public void onClick(View v) {
            Intent intent = new Intent(getContext(), EditRoverDataLinkParamActivity.class);
            intent.putExtra("DataLinkType", dataLinkParameter.Type.swigValue());
            switch (dataLinkParameter.Type)
            {
               case ExtendSource:
               {
                  intent.putExtra("BluetoothParam",
dataLinkParameter.bluetoothParam.toString());
               }
                  break;
            }
            startActivityForResult(intent, R.id.linearLayout_DataLink&0xFFFF);
         }
      });
   }
   private void initializeMountPointList()
   {
      final CustomTextViewLayoutSelectEdit MountPoint =
mInfoView.findViewById(R.id.layoutSelectEdit_MountPoint);
      MountPoint.setCustomClickListener(new OnClickListener() {
         @Override
         public void onClick(View v) {
            CustomInputActivity.InputListener inputListener = new
CustomInputActivity.InputListener() {
               @Override
               public void onSelect(String label, int index) {
                  if (label.indexOf('(') > 0&&
                        label.indexOf(')') > 0) {
                     label = label.substring(0, label.indexOf('('));
                  }
                  MountPoint.setText(label);
               }
            };
            CustomInputActivity.SortChangeListener sortChangeListener = new
CustomInputActivity.SortChangeListener() {
               @Override
               public void onSortTypeChange(int index) {
                  SourecTableManage.getInstance().setSortType(index);
               }
               @Override
               public ArrayList<String> getValueList() {
                  return SourecTableManage.getInstance().getMountPointList();
               }
               @Override
               public String getValue(String inputOld) {
                  return inputOld;
               }
               @Override
               public ArrayList<String> getSortTypeList() {
```

```
                ArrayList<String> listSort = new ArrayList<>();
                listSort.add(getString(R.string.label_sort_name_defalt));
                listSort.add(getString(R.string.label_sort_name_pos));
                listSort.add(getString(R.string.label_sort_name_neg));
                if (SourecTableManage.getInstance().isSurportSortByDist()) {
                   listSort.add(getString(R.string.label_sort_dist_pos));
                }
                return listSort;
             }
             @Override
             public int getSortIndex() {
                return SourecTableManage.getInstance().getSortType();
             }
          };
          new CustomInputActivity.Builder(getContext())
                .setMode(CustomInputActivity.MODE_EDIT_SELECT)
                .setList(SourecTableManage.getInstance().getMountPointList())
                .setValue(MountPoint.getText())
                .setTitle(getString(R.string.string_ntrip_mount_point))
                .setListener(inputListener)
                .setSortChangeListener(sortChangeListener)
                .start();
       }
    });
}
@Override
public void onClick(View v)
 {
    switch(v.getId())
    {
       case R.id.button_MountPoint:
       {
          onButtonGetMountPoint();
       }
          break;
       case R.id.button_Start:
       {
          onButtonStart();
       }
       break;
       case R.id.button_Stop:
       {
          onButtonStop();
       }
       break;
    }
}
@Override
public void onActivityResult(int requestCode,int resultCode,Intent data)
{
```

```
        super.onActivityResult(requestCode, resultCode, data);
        if(resultCode == AppFinalUtil.RETURN_CODE_VALUE_BACK)
        {
              if (requestCode == (R.id.linearLayout_DataLink&0xFFFF) && data != null)
           {
              switch (dataLinkParameter.Type){
                 case ExtendSource:
dataLinkParameter.bluetoothParam.parseValue(data.getStringExtra("BluetoothParam"));
                    dataLinkParameter.bluetoothParam.strMountPoint =
getTextViewString(R.id.layoutSelectEdit_MountPoint);
                    break;
              }
              updateValueToUI();
           }
        }
     }
     private void onButtonGetMountPoint() {
        if (mInfoView == null)
           return;
        mnTimeCount = 0;
        mHandler.postDelayed(mRunnable, 1000);
        tagConnectionParameter connectionParameter = new tagConnectionParameter();
        connectionParameter.set(dataLinkParameter.bluetoothParam);
        CorsClientManage.GetInstance().setNetworkParameter(connectionParameter.connectMode,
connectionParameter.strIP,      connectionParameter.nPort,      connectionParameter.strUser,
connectionParameter.strPassword);
        CorsClientManage.GetInstance().onUpdateMountPoint();
        CustomWaittingLayout                    waittingLayout =
AppFinalUtil.activityCurrent.findViewById(R.id.waittingLayout);
        if (waittingLayout != null) {
waittingLayout.setLabel(getString(R.string.command_function_set_start_get_mount_point));
           waittingLayout.setVisibility(View.VISIBLE);
        }
     }
     public void onButtonStart() {
        ConfigNtripParameter.getInstance().setAutoConnect(true);
        if(ConnectListener.CommanderStatus.SUCCESS !=
DeviceManage.GetInstance().getConnectedStatus())
        {
           Toast toast = Toast.makeText(getContext(),
R.string.string_prompt_communication_not_connected, Toast.LENGTH_SHORT);
           toast.setGravity(Gravity.CENTER, 0, 0);
           toast.show();
           return;
        }
        if (CorsClientManage.GetInstance().isConnectCors())
        {
           return;
        }
        dataLinkParameter.bluetoothParam.strMountPoint =
```

```
getTextViewString(R.id.layoutSelectEdit_MountPoint);
      ConfigNtripParameter.getInstance().set(dataLinkParameter.bluetoothParam);
      ConfigNtripParameter.getInstance().saveConfig();
CorsClientManage.GetInstance().setNetworkParameter(dataLinkParameter.bluetoothParam.getMode
(eDeviceWorkingMode.Rover),
            dataLinkParameter.bluetoothParam.strIP, dataLinkParameter.bluetoothParam.nPort,
            dataLinkParameter.bluetoothParam.strUser,
dataLinkParameter.bluetoothParam.strPassword,
            dataLinkParameter.bluetoothParam.strMountPoint);
      CorsClientManage.GetInstance().startConnectCors();
      updateConnectUI();
   }
   public void onButtonStop()
   {
      ConfigNtripParameter.getInstance().setAutoConnect(false);
      if (!CorsClientManage.GetInstance().isConnectCors())
      {
         return;
      }
      CorsClientManage.GetInstance().stopConnectCors();
      updateConnectUI();
   }
   public void updateConnectUI()
   {
      if (mInfoView == null)
         return;
      View viewCors = mInfoView.findViewById(R.id.linearLayout_Cors);
      boolean bConnect = CorsClientManage.GetInstance().isConnectCors();
      if (viewCors.getVisibility() != View.VISIBLE)
      {
         bConnect = false;
      }
      setVisibility(R.id.button_Start, bConnect?View.GONE:View.VISIBLE);
      setVisibility(R.id.button_Stop, bConnect?View.VISIBLE:View.GONE);
      setEnabled(R.id.button_MountPoint, !bConnect);
      setEnabled(R.id.layoutSelectEdit_MountPoint, !bConnect);
      setEnabled(R.id.layoutSelect_DataLink, false/*!bConnect*/);
      setEnabled(R.id.linearLayout_DataLink, !bConnect);
   }
   int mnCountReceiver = 0;
   int nLength = 0;
   long currentTime = 0;
   public void receiverNetworkData(int length)
   {
      if (currentTime/1000 == System.currentTimeMillis()/1000)
      {
         nLength += length;
      }
      else {
         currentTime = System.currentTimeMillis();
```

```
            nLength = length;
        }
        Message message = new Message();
        Bundle bundle = new Bundle();
        bundle.putInt("length", nLength);
        message.what = 0;
        message.setData(bundle);
        mHandler.sendMessage(message);
    }
    public void updateMountPointFinish()
    {
        mnTimeCount = 61;
    }
    private Handler mHandler = new Handler()
    {
        public void handleMessage(Message msg)
        {
            switch (msg.what)
            {
                case 0: {
                    View viewCors = mInfoView.findViewById(R.id.linearLayout_Cors);
                    if (viewCors.getVisibility() != View.VISIBLE)
                        return;
                    int length = msg.getData().getInt("length");
                    CustomTimerView timerView_Rev = mInfoView.findViewById(R.id.timerView_Rev);
                    mnCountReceiver += length;
                    mnCountReceiver = (mnCountReceiver%8000);
                    timerView_Rev.setPosValue(mnCountReceiver);
                    timerView_Rev.setPromptTextString(String.format("%dB", length));
                }
                break;
            }
        }
    };
    int mnTimeCount = 0;
    Runnable mRunnable = new Runnable()
    {
        @Override
        public void run()
        {
            mnTimeCount++;
            if (mnTimeCount == 60) {
                DeviceUpdateUINotify.eventUpdateMountPointSuccess(false);
            }
            if (mnTimeCount < 60) {
                mHandler.postDelayed(this, 1000);
            }
        }
    };
    @Override
```

```
    public void onDestroy() {
        DeviceManage.GetInstance().setPageInBlueToothSetting(false);
        mHandler.removeCallbacks(mRunnable);
        super.onDestroy();
    }
}
public class ProjectCreateActivity extends CommonEventBaseFragmentActivity
        implements OnClickListener, ViewPager.OnPageChangeListener
{
    private ProjectBasicInfoFragment basicInfoFragment = null;
    private ProjectCoordinateSystemFragment coordinateSystemFragment = null;
    private CommonFragmentAdapter mFragmentAdapter;
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_project_create);
        initialUI();
    }
    private void initialUI()
    {
        initClickListener(R.id.button_OK, this);
        initClickListener(R.id.button_Pre, this);
        initClickListener(R.id.button_Next, this);
        initClickListener(R.id.button_Cancel, this);
        mFragmentAdapter = new CommonFragmentAdapter(getSupportFragmentManager());
        basicInfoFragment = new ProjectBasicInfoFragment();
        coordinateSystemFragment = new ProjectCoordinateSystemFragment();
        mFragmentAdapter.addFragment(basicInfoFragment);
        mFragmentAdapter.addFragment(coordinateSystemFragment);
        ViewPager viewPager = findViewById(R.id.viewPager_Fragment);
        viewPager.setAdapter(mFragmentAdapter);
        viewPager.addOnPageChangeListener(this);
        TabLayout tabLayout = findViewById(R.id.tabLayout_Fragment);
        tabLayout.setupWithViewPager(viewPager);
    }
    @Override
    public void onPageScrolled(int i, float v, int i1) {
        setVisibility(R.id.button_Cancel, i==0?View.VISIBLE:View.GONE);
        setVisibility(R.id.button_Next, i==0?View.VISIBLE:View.GONE);
        setVisibility(R.id.button_Pre, i==1?View.VISIBLE:View.GONE);
        setVisibility(R.id.button_OK, i==1?View.VISIBLE:View.GONE);
        if (basicInfoFragment == null || coordinateSystemFragment == null)
            return;
        if (i==1)
        {
            if (basicInfoFragment.isUseLastCoordinateSystemParam())
            {
coordinateSystemFragment.setCoordSystemType(ProjectManage.GetInstance().getUseCoordSystemTy
pe());
```

```
            if (coordinateSystemFragment.getCoordSystemType() ==
eCoordinateSystemType.SYSTEM_TYPE_LOCAL)
            {
coordinateSystemFragment.setCoordinateSystemParameter(ProjectCoordSystem.GetInstance().getC
oordSystemParameter());
            }
            else if (coordinateSystemFragment.getCoordSystemType() ==
eCoordinateSystemType.SYSTEM_TYPE_RTCM)
            {
coordinateSystemFragment.setRtcmCoordinateSystemParameter(ProjectCoordSystem.GetInstance().
getRtcmCoordSystemParameter());
            }
         }
         else {
coordinateSystemFragment.setCoordSystemType(eCoordinateSystemType.SYSTEM_TYPE_LOCAL);
            tagCoordinateSystemParameter systemParameter = new tagCoordinateSystemParameter();
            if(AppFinalUtil.isLanguageZhCh())
            {
               systemParameter.setName("CGCS2000");
               tagEllipsoidParameter ellipsoidParameter = new tagEllipsoidParameter();
               ellipsoidParameter.setName("CGCS2000");
               ellipsoidParameter.setDf(298.257222101);
               ellipsoidParameter.setDa(6378137);
               systemParameter.setEllipsoidParam(ellipsoidParameter);
            }
            else {
               systemParameter.setName("Default");
            }
            coordinateSystemFragment.setCoordinateSystemParameter(systemParameter);
         }
      }
   }
   @Override
   public void onPageSelected(int i) {
   }
   @Override
   public void onPageScrollStateChanged(int i) {
   }
   @Override
   public void onClick(View v) {
      switch (v.getId()) {
         case R.id.button_OK: {
            if (!basicInfoFragment.testProjectName())
            {
               ViewPager viewPager = findViewById(R.id.viewPager_Fragment);
               viewPager.setCurrentItem(0);
               basicInfoFragment.setTextViewFocus(R.id.editText_ProjectName);
               return;
            }
            if (!createProject())
```

```
            {
                return;
            }
            setResult(AppFinalUtil.RETURN_CODE_DEFAULT);
            finish();
        }
        break;
        case R.id.button_Pre:
        {
            ViewPager viewPager = findViewById(R.id.viewPager_Fragment);
            viewPager.setCurrentItem(0);
        }
        break;
        case R.id.button_Next:
        {
            ViewPager viewPager = findViewById(R.id.viewPager_Fragment);
            viewPager.setCurrentItem(1);
        }
        break;
        case R.id.button_Cancel:
        {
            finish();
        }
        break;
        default:
            break;
    }
}
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    ViewPager viewPager = findViewById(R.id.viewPager_Fragment);
mFragmentAdapter.getItem(viewPager.getCurrentItem()).onActivityResult(requestCode&0xFFFF,
resultCode, data);
}
private boolean createProject()
{
    String strProjectPath = basicInfoFragment.getProjectPath();
    String strProjectName = basicInfoFragment.getProjectName();
    String strOperator = basicInfoFragment.getOperator();
    String strRemark = basicInfoFragment.getRemark();
    String strCreateTime = basicInfoFragment.getCreateTime();
    eCoordinateSystemType systemType = coordinateSystemFragment.getCoordSystemType();
    tagCoordinateSystemParameter coordinateSystemParameter =
coordinateSystemFragment.getCoordinateSystemParameter();
    BaseChangeCorrectParameter correctParameter = new BaseChangeCorrectParameter();
    tagGnssRefStationItem refStationItem = null;
    if (basicInfoFragment.isRetainBaseCorrectParam())
    {
        correctParameter.set(ConfigBaseCorrect.getInstance());
        refStationItem =
```

```
ProjectDbFileManage.getInstance().getRefStationData(correctParameter.strBaseKeyId);
      }
      if (!ProjectManage.GetInstance().createProject(strProjectPath, strProjectName,
            strOperator, strRemark, strCreateTime, systemType, coordinateSystemParameter))
      {
         return false;
      }
      if (systemType == eCoordinateSystemType.SYSTEM_TYPE_RTCM)
      {
ProjectCoordSystem.GetInstance().setRtcmCoordSystemParameter(coordinateSystemFragment.getRt
cmCoordinateSystemParameter());
      }
      if (refStationItem != null)
      {
         ProjectDbFileManage.getInstance().insertRefStationInfo(refStationItem);
      }
      if (correctParameter.isValid())
      {
         ProjectDbFileManage.getInstance().insertBaseCorrectParameter(correctParameter);
      }
      if (!correctParameter.isEquals(ConfigBaseCorrect.getInstance())) {
         ConfigBaseCorrect.getInstance().set(correctParameter);
         ConfigBaseCorrect.getInstance().saveConfig();
      }
      return true;
   }
    public void onEventMainThread(MainEvent.UpdateRtcmParam obj) {
        tagRtcmCoordinateSystemParameter            coordinateSystemParameter            =
coordinateSystemFragment.getRtcmCoordinateSystemParameter();
        tagRtcmCoordinateSystemParameter               rtcmSystemParameter               =
ProjectCoordSystem.GetInstance().getRtcm31Convert().getRtcmCoordinateSystemParameter();
        if (coordinateSystemParameter.getUseSrcEllipsoidRtcm())
      {
coordinateSystemParameter.setSrcEllipsoidVaild(rtcmSystemParameter.getSrcEllipsoidVaild());
coordinateSystemParameter.setSrcEllipsoidParam(rtcmSystemParameter.getSrcEllipsoidParam());
      }
      if (coordinateSystemParameter.getUseTargetEllipsoidRtcm())
      {
coordinateSystemParameter.setTargetEllipsoidVaild(rtcmSystemParameter.getTargetEllipsoidVai
ld());
coordinateSystemParameter.setTargetEllipsoidParam(rtcmSystemParameter.getTargetEllipsoidPar
am());
      }
      if (coordinateSystemParameter.getUseProjectRtcm())
      {
         coordinateSystemParameter.setProjectVaild(rtcmSystemParameter.getProjectVaild());
         coordinateSystemParameter.setProjectParam(rtcmSystemParameter.getProjectParam());
      }
      if (coordinateSystemParameter.getUseServenRtcm())
      {
```

```
            coordinateSystemParameter.setServenVaild(rtcmSystemParameter.getServenVaild());
            coordinateSystemParameter.setDatumParam(rtcmSystemParameter.getDatumParam());
        }
        if (coordinateSystemParameter.getUseGridRtcm())
        {
            coordinateSystemParameter.setGridVaild(rtcmSystemParameter.getGridVaild());
            coordinateSystemParameter.setGridParam(rtcmSystemParameter.getGridParam());
        }
coordinateSystemFragment.setRtcmCoordinateSystemParameter(coordinateSystemParameter);
        coordinateSystemFragment.updateValueToUI();
     }
}
public class ProjectManage extends ProjectConfig
{
    private static ProjectManage mProjectManage = null;
    public static ProjectManage GetInstance()
    {
        if (null == mProjectManage)
        {
            mProjectManage = new ProjectManage();
            mProjectManage.initializeXSurvey();
        }
        return mProjectManage;
    }
    private void initializeXSurvey()
    {
        mstrProgramPath = getDefaultProgramPath();
        loadProjectConfig();
        IoFileManage.createDirectory(mstrProgramPath);
        mstrLastProjectPath = getProgramProjectPath();
        IoFileManage.createDirectory(mstrLastProjectPath);
        IoFileManage.createDirectory(getProgramConfigPath());
        IoFileManage.createDirectory(getProgramDebugPath());
        IoFileManage.createDirectory(getProgramImportPath());
        IoFileManage.createDirectory(getProgramExportPath());
        IoFileManage.createDirectory(getProgramMapPath());
        IoFileManage.createDirectory(getProgramDataPath());
        IoFileManage.createDirectory(getProgramGeoidPath());
        IoFileManage.createDirectory(getProgramInstallationPath());
    }
    public boolean loadLastProject()
    {
        if (mProjectItemList.size() > 0)
            return openProject(mProjectItemList.get(0).strProjectFilePath);
        return false;
    }
    public void removeProjectItem(String strProjectFilePath)
    {
        for (int i = mProjectItemList.size()-1; i >= 0; i--)
        {
```

```
            if (mProjectItemList.get(i).strProjectFilePath.equalsIgnoreCase(strProjectFilePath))
            {
               mProjectItemList.remove(i);
               saveLastProjectName();
            }
         }
      }
      public String getProjectPath()
      {
         return mstrLastProjectPath;
      }
      public boolean createProject(String strProjectPath, String strProjectName, String
strOperator, String strRemark, String strCreateTime,
                              eCoordinateSystemType systemType, tagCoordinateSystemParameter
systemParameter)
      {
         File temFile = new File(strProjectPath + "/" + strProjectName);
         if (temFile.exists())
         {
            return false;
         }
         IoFileManage.createDirectory(strProjectPath + "/" + strProjectName);
         closeProject();
         setOperator(strOperator);
         setRemark(strRemark);
         setCreateTime(strCreateTime);
         String strProjectPathName = strProjectPath + "/" + strProjectName + "/config.job";
         if (!saveConfig(strProjectPathName))
         {
            return false;
         }
         misOpenProject = true;
         mstrLastProjectPath = strProjectPath;
         mstrProjectName = strProjectName;
         IoFileManage.createDirectory(getProjectDataDirectory());
         IoFileManage.createDirectory(getProjectConfigDirectory());
         String strDataFilePath = getProjectDirectory() + "/" + "surveydata.db";
         ProjectDbFileManage.getInstance().createSQLiteDB(strDataFilePath);
         setUseCoordSystemType(systemType);
         ProjectCoordSystem.GetInstance().setConfigPath(getProjectDirectory()    +    "/"    +
"coordparam.sys");
         if (systemType == eCoordinateSystemType.SYSTEM_TYPE_LOCAL) {
            ProjectCoordSystem.GetInstance().setCoordSystemParameter(systemParameter);
            ProjectCoordSystem.GetInstance().Save();
         }
         loadConfig();
         ConfigConvertParam.getInstance().initConfig();
         ConfigStakeoutPoint.getInstance().initConfig();
         tagProjectItem projectItem = new tagProjectItem();
         projectItem.strProjectName = strProjectName;
```

```
        projectItem.strProjectFilePath = strProjectPathName;
        projectItem.strLastTime     =     StringFunction.formatDate("yyyy-MM-dd     HH:mm:ss",
Calendar.getInstance().getTime());
        projectItem.strCreateTime = getCreateTime();
        projectItem.strOperator = getOperator();
        if (mProjectItemList.size() > 0)
           mProjectItemList.add(0, projectItem);
        else
           mProjectItemList.add(projectItem);
        saveLastProjectName();
        return true;
     }
     public boolean renameProject(String strProjectName)
     {
        String strTemProjectName = getProjectName();
        File temFile = new File(getProjectPath() + "/" + strProjectName);
        if (temFile.exists())
        {
           return false;
        }
        ProjectDbFileManage.getInstance().closeSQLiteDB();
        File oldFile = new File(getProjectPath() + "/" + strTemProjectName);
        oldFile.renameTo(temFile);
        mstrProjectName = strProjectName;
        oldFile    =    new    File(StringFunction.format("%s/%s.bck",    getProgramDataPath(),
strTemProjectName));
        if (oldFile.exists()){
           temFile   =   new   File(StringFunction.format("%s/%s.bck",   getProgramDataPath(),
strProjectName));
           oldFile.renameTo(temFile);
        }
        String strDataFilePath = getProjectDirectory() + "/" + "surveydata.db";
        ProjectDbFileManage.getInstance().openSQLiteDB(strDataFilePath);
        String strProjectPathName = getProjectPath() + "/" + strTemProjectName + "/config.job";
        for (int i = mProjectItemList.size()-1; i >= 0; i--)
        {
           if (mProjectItemList.get(i).strProjectFilePath.equalsIgnoreCase(strProjectPathName))
           {
              tagProjectItem projectItem = mProjectItemList.get(i);
              strProjectPathName = getProjectPath() + "/" + strProjectName + "/config.job";
              projectItem.strProjectName = mstrProjectName;
              projectItem.strProjectFilePath = strProjectPathName;
              projectItem.strLastTime   =   StringFunction.formatDate("yyyy-MM-dd   HH:mm:ss",
Calendar.getInstance().getTime());
              saveLastProjectName();
           }
        }
        return true;
     }
     public boolean openProject(String strProjectPathName)
```

```
    {
        if (!loadConfig(strProjectPathName))
            return false;
        strProjectPathName = strProjectPathName.replace("//", "/");
        String strTem = strProjectPathName.substring(0, strProjectPathName.lastIndexOf('/'));
        mstrLastProjectPath = strTem.substring(0, strTem.lastIndexOf('/'));
        mstrProjectName = strTem.substring(strTem.lastIndexOf('/') + 1);
        IoFileManage.createDirectory(getProjectDataDirectory());
        String strDataFilePath = getProjectDirectory() + "/" + "surveydata.db";
        ProjectDbFileManage.getInstance().openSQLiteDB(strDataFilePath);
        setUseEncrypt(isCoordinateSystemParEncrypt());
        loadConfig();
        ProjectCoordSystem.GetInstance().setUseCoordSystemType(getUseCoordSystemType());
        IoFileManage.createDirectory(getProjectConfigDirectory());
        removeProjectItem(strProjectPathName);
        tagProjectItem projectItem = new tagProjectItem();
        projectItem.strProjectName = mstrProjectName;
        projectItem.strProjectFilePath = strProjectPathName;
        projectItem.strLastTime     =     StringFunction.formatDate("yyyy-MM-dd     HH:mm:ss",
Calendar.getInstance().getTime());
        projectItem.strCreateTime = getCreateTime();
        projectItem.strOperator = getOperator();
        if (mProjectItemList.size() > 0)
            mProjectItemList.add(0, projectItem);
        else
            mProjectItemList.add(projectItem);
        saveLastProjectName();
        misOpenProject = true;
        return true;
    }
    public void setUseEncrypt(boolean bEncrypt)
    {
        setCoordinateSystemParEncrypt(bEncrypt);
        Save();
        ProjectCoordSystem.GetInstance().loadConfig(getProjectDirectory() + "/coordparam.sys");
    }
    public void setUseCoordSystemType(eCoordinateSystemType nType) {
        super.setUseCoordSystemType(nType);
        if (isOpenProject()) {
            Save();
        }
        ProjectCoordSystem.GetInstance().setUseCoordSystemType(nType);
    }
    public boolean InputPassword(String strPassword)
    {
        return ProjectCoordSystem.GetInstance().inputPassword(strPassword);
    }
    @Override
    public void setOperator(String strOperator) {
        if (isOpenProject())
```

```
        {
            if (mProjectItemList.size() > 0)
                mProjectItemList.get(0).strOperator = strOperator;
            saveLastProjectName();
        }
        super.setOperator(strOperator);
    }
    public void Save()
    {
        String strProjectPathName = mstrLastProjectPath + "/" + mstrProjectName + "/config.job";
        saveConfig(strProjectPathName);
    }
    private void loadConfig()
    {
        ConfigConvertParam.getInstance().loadConfig();
        ConfigRecord.getInstance().loadConfig();
        ConfigMapConvert.getInstance().loadConfig();
        ConfigCadEdit.getInstance().loadConfig();
        ConfigStakeoutPoint.getInstance().loadConfig();
        ConfigDisplayInfoManage.getInstance().loadConfig();
    }
    public void closeProject()
    {
        String strDataFilePath = getProjectDirectory() + "/" + "surveydata.db";
        ProjectDbFileManage.getInstance().closeSQLiteDB();
        MediaScannerConnection.scanFile(AppFinalUtil.activityCurrent,
                new String[] { strDataFilePath }, null, null);
        misOpenProject = false;
        mstrProjectName = "";
        clear();
    }
    public String getProjectName() {
        return mstrProjectName;
    }
    public boolean isOpenProject()
    {
        return misOpenProject;
    }
    public ArrayList<tagProjectItem> getProjectItemList()
    {
        return mProjectItemList;
    }
    public String getProjectDirectory()
    {
        return getProjectPath() + "/" + mstrProjectName;
    }
    public String getProjectDataDirectory()
    {
        return getProjectDirectory() + "/Data";
    }
```

```
    public String getProjectConfigDirectory()
    {
       return getProjectDirectory() + "/Config";
    }
    public String getProjectGisImagesDirectory() {
       return getProjectDirectory() + "/Images";
    }
     public String getProjectGisFileDirectory() {
         return getProjectDirectory() + "/Files";
     }
    public String getDefaultProgramPath()
    {
       String strPathProgram = Environment.getExternalStorageDirectory().getAbsolutePath() +
AppFinalUtil.getAppID().getAppDirectory();
       return strPathProgram;
    }
    public String getProgramProjectPath()
    {
       String strProjectPath = getProjectPath();
       if (strProjectPath == null || strProjectPath.length() <= 0)
       {
          strProjectPath = mstrProgramPath + "/Project";
       }
       return strProjectPath;
    }
    public String getProgramConfigPath()
    {
       return mstrProgramPath+"/Config";
    }
    public String getProgramDebugPath()
    {
       return mstrProgramPath+"/Debug";
    }
    public String getProgramDataPath()
    {
       return mstrProgramPath+"/Data";
    }
    public String getProgramImportPath()
    {
       return mstrProgramPath+"/Import";
    }
    public String getProgramExportPath()
    {
       return mstrProgramPath+"/Export";
    }
    public String getProgramMapPath()
    {
       return mstrProgramPath+"/Map";
    }
    public String getProgramGeoidPath() {
```

```
        return mstrProgramPath + "/Geoid";
    }
    public String getProgramInstallationPath() {
        return mstrProgramPath + "/Installation";
    }
    public String getProgramCachePath() {
        return mstrProgramPath + "/.Cache";
    }
    private void loadProjectConfig()
    {
        mProjectItemList.clear();
        String programConfigFile = getProgramConfigPath() + "/projectInfo.ini";
        CommonFile temFile = new CommonFile(programConfigFile);
        CBufferData temBufferData = temFile.openReadAll(CommonFile.FORMAT_UTF8);
        if (temBufferData == null)
            return;
        CParseString temParseString = new CParseString();
        String strLine;
        while ((strLine = temBufferData.ReadLine()) != null) {
            if (strLine.isEmpty())
                continue;
            if(strLine.contains(", "))
                strLine = strLine.replaceAll(", ", ",");
            int nCount = temParseString.split(strLine, ",");
            if (nCount < 4) {
                continue;
            }
            tagProjectItem item = new tagProjectItem();
            item.strProjectName = temParseString.getValueString(0);
            item.strProjectFilePath = temParseString.getValueString(1);
            item.strLastTime = temParseString.getValueString(2);
            item.strCreateTime = temParseString.getValueString(3);
            item.strOperator = temParseString.getValueString(4);
            mProjectItemList.add(item);
        }
    }
    private void saveLastProjectName()
    {
        String programConfigFile = getProgramConfigPath() + "/projectInfo.ini";
        CommonFile temFile = new CommonFile(programConfigFile);
        if(temFile.isExist())
        {
            temFile.delete();
        }
        if (temFile.openWrite())
        {
            tagProjectItem item;
            for (int i = 0; i < mProjectItemList.size(); i++)
            {
                item = mProjectItemList.get(i);
```

```
            String strLine = StringFunction.format("%s,%s,%s,%s,%s\r\n", item.strProjectName,
item.strProjectFilePath, item.strLastTime, item.strCreateTime, item.strOperator);
            temFile.write(strLine);
         }
         temFile.close();
      }
   }
   private boolean misOpenProject = false;
   private String mstrLastProjectPath = "";
   private String mstrProjectName = "";
   private String mstrProgramPath = "";
   private ArrayList<tagProjectItem> mProjectItemList = new ArrayList<>();
}
public class PointLibraryActivityV2 extends CommonBaseFragmentActivity implements
PointCommonFragment.OnChangeListener{
   public static ArrayList<Long> cSelectPointIdItems = null;
   private ePointLibraryMode pointLibraryMode = ePointLibraryMode.MODE_NULL;
   private StakePointPreviewFragment pointPreviewFragment = null;
   private CommonFragmentAdapter mFragmentAdapter;
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_point_library_v2);
      int mode = getIntent().getIntExtra("PointLibraryMode", -1);
      pointLibraryMode = ePointLibraryMode.swigToEnum(mode);
      initClickListener(R.id.imageButton_Mode, new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            ViewPager viewPager = findViewById(R.id.viewPager_Fragment);
            CommonV4Fragment commonV4Fragment =
mFragmentAdapter.getItem(viewPager.getCurrentItem());
            if (commonV4Fragment instanceof PointListCommonFragment) {
               boolean bDisplayList =
((PointListCommonFragment)commonV4Fragment).isDisplayList();
               ((PointListCommonFragment)commonV4Fragment).setDisplayMode(!bDisplayList);
               ImageView imageButtonMode = findViewById(R.id.imageButton_Mode);
imageButtonMode.setImageResource(bDisplayList?R.drawable.icon_point_mode_list:R.drawable.ic
on_point_mode_grid);
            }
         }
      });
      mFragmentAdapter = new CommonFragmentAdapter(getSupportFragmentManager());
      switch (pointLibraryMode)
      {
         case MODE_NULL:
            mFragmentAdapter.addFragment(new PointLibraryFragment(this));
            break;
         case MODE_POINT_STAKEOUT: {
            setTitle(getString(R.string.title_point_stakeout));
            boolean bReturnToSurveyMain =
getIntent().getBooleanExtra("ReturnToSurveyMain",false);
```

```
            PointLibraryFragment pointFragment = new PointLibraryFragment(this);
            pointFragment.setStakeoutReturnToMain(bReturnToSurveyMain);
            PointLibraryStakePointFragment stakePointFragment = new
PointLibraryStakePointFragment(this);
            stakePointFragment.setStakeoutReturnToMain(bReturnToSurveyMain);
            mFragmentAdapter.addFragment(pointFragment);
            mFragmentAdapter.addFragment(stakePointFragment);
            pointPreviewFragment = new StakePointPreviewFragment();
            mFragmentAdapter.addFragment(pointPreviewFragment);
            break;
        }
        case MODE_SELECT_POINT:{
            setTitle(getString(R.string.title_coordinate_select));
            PointLibraryFragment pointFragment = new PointLibraryFragment(this);
            pointFragment.setSimpleSelect(true);
            mFragmentAdapter.addFragment(pointFragment);
            mFragmentAdapter.addFragment(new PointLibraryInputFragment(this));
            break;
        }
        case MODE_SELECT_INPUT_POINT:{
            setTitle(getString(R.string.title_coordinate_select));
            PointLibraryFragment pointFragment = new PointLibraryFragment(this);
            pointFragment.setSimpleSelect(true);
            mFragmentAdapter.addFragment(pointFragment);
            mFragmentAdapter.addFragment(new PointLibraryInputFragment(this));
            break;
        }
        case MODE_SELECT_STAKE_POINT_LIST: {
            setTitle(getString(R.string.title_coordinate_batch_select));
            mFragmentAdapter.addFragment(new PointLibrarySelectStakeListFragment(this));
            break;
        }
        case MODE_SELECT_POINT_LIST: {
            setTitle(getString(R.string.title_coordinate_batch_select));
            mFragmentAdapter.addFragment(new PointLibrarySelectListFragment(this));
            break;
        }
        case MODE_SELECT_RESTORE_DELETE_POINT: {
            setTitle(getString(R.string.title_restore_data));
            mFragmentAdapter.addFragment(new PointLibraryRestoreDeletePointFragment(this));
            break;
        }
        default: {
            mFragmentAdapter.addFragment(new PointLibraryFragment(this));
            break;
        }
    }
    ViewPager viewPager = findViewById(R.id.viewPager_Fragment);
    viewPager.setAdapter(mFragmentAdapter);
    TabLayout tabLayout = findViewById(R.id.tabLayout_Fragment);
```

```
        tabLayout.setupWithViewPager(viewPager);
        if (mFragmentAdapter.getCount() <= 1)
            tabLayout.setVisibility(View.GONE);
        switch (pointLibraryMode)
        {
            case MODE_POINT_STAKEOUT:
            case MODE_SELECT_INPUT_POINT:
                viewPager.setCurrentItem(1);
                break;
        }
        {
            viewPager.addOnPageChangeListener(new ViewPager.OnPageChangeListener() {
                @Override
                public void onPageScrolled(int i, float v, int i1) {
                    NoScrollViewPager viewPager = findViewById(R.id.viewPager_Fragment);
                    CommonV4Fragment                        commonV4Fragment                        =
mFragmentAdapter.getItem(viewPager.getCurrentItem());
                    if (pointPreviewFragment != null) {
                        viewPager.setNoScroll(commonV4Fragment == pointPreviewFragment);
                        if (commonV4Fragment == pointPreviewFragment) {

pointPreviewFragment.setPointItems(PointLibraryManage.getInstance().getStakePointList());
                        }
                    }
                    if (commonV4Fragment instanceof PointListCommonFragment)
                    {
                        setVisibility(R.id.linearLayout_DisplayMode, View.VISIBLE);
                    }
                    else {
                        setVisibility(R.id.linearLayout_DisplayMode, View.GONE);
                    }
                    commonV4Fragment.updateValueToUI();
                }
                @Override
                public void onPageSelected(int i) {
                }
                @Override
                public void onPageScrollStateChanged(int i) {
                }
            });
        }
    }
    @Override
    public void finish()
    {
        ViewPager viewPager = findViewById(R.id.viewPager_Fragment);
        CommonV4Fragment                              commonV4Fragment                              =
mFragmentAdapter.getItem(viewPager.getCurrentItem());
        if (commonV4Fragment instanceof PointCommonFragment) {
            if                                                          (((PointCommonFragment)
```

```
mFragmentAdapter.getItem(viewPager.getCurrentItem())).cancelBatchSelectMode())
            return;
        }
        super.finish();
    }
    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        ViewPager viewPager = findViewById(R.id.viewPager_Fragment);
mFragmentAdapter.getItem(viewPager.getCurrentItem()).onActivityResult(requestCode&0xFFFF,
resultCode, data);
    }
    @Override
    public void onChanged() {
        ViewPager viewPager = findViewById(R.id.viewPager_Fragment);
        for (int i=0; i<mFragmentAdapter.getCount(); i++)
        {
            if (i!= viewPager.getCurrentItem())
            {
                CommonV4Fragment commonV4Fragment = mFragmentAdapter.getItem(i);
                if (commonV4Fragment instanceof PointCommonFragment) {
                    ((PointCommonFragment)commonV4Fragment).setDataReLoad();
                }
            }
        }
    }
    @Override
    public void importFileFinish(boolean success) {
        mHandler.sendEmptyMessage(success?0:1);
    }
    @Override
    public void showWaitingDialog(boolean isShow)
    {
        mHandler.sendEmptyMessage(isShow?5:6);
    }
    @Override
    public void promptRepetitionName() {
        mHandler.sendEmptyMessage(8);
    }
    @Override
    public void searchFinish() {
        mHandler.sendEmptyMessage(7);
    }
    @SuppressLint("HandlerLeak")
    Handler mHandler = new Handler()
    {
        public void handleMessage(Message msg)
        {
            switch (msg.what)
            {
                case 0: {
```

```
                makeTextString(R.string.string_prompt_import_file_succeed);
                setVisibility(R.id.waittingLayout, View.GONE);
                ViewPager viewPager = findViewById(R.id.viewPager_Fragment);
                CommonV4Fragment commonV4Fragment =
mFragmentAdapter.getItem(viewPager.getCurrentItem());
                if (commonV4Fragment instanceof PointCommonFragment)
                   ((PointCommonFragment)commonV4Fragment).setDataReLoad();
                commonV4Fragment.updateValueToUI();
                onChanged();
             }
                break;
             case 1:
                makeTextString(R.string.string_prompt_import_file_failed);
                setVisibility(R.id.waittingLayout, View.GONE);
                break;
             case 2: {
                makeTextString(R.string.string_prompt_export_file_succeed);
                setVisibility(R.id.waittingLayout, View.GONE);
             }
             break;
             case 3:
                makeTextString(R.string.string_prompt_export_file_failed);
                setVisibility(R.id.waittingLayout, View.GONE);
                break;
             case 5:
                setVisibility(R.id.waittingLayout, View.VISIBLE);
                break;
             case 6:
                setVisibility(R.id.waittingLayout, View.GONE);
                break;
             case 7:
             {
                ViewPager viewPager = findViewById(R.id.viewPager_Fragment);
                CommonV4Fragment commonV4Fragment =
mFragmentAdapter.getItem(viewPager.getCurrentItem());
                commonV4Fragment.updateValueToUI();
                setVisibility(R.id.waittingLayout, View.GONE);
             }
                break;
             case 8:
             {
                setVisibility(R.id.waittingLayout, View.GONE);
                new CustomDialog(AppFinalUtil.activityCurrent, R.string.string_prompt
                      , R.string.string_prompt_point_name_repetition_prompt
                      , R.string.button_continue, R.string.button_cancel)
                      .setDialogListener(new CustomDialog.OnDialogListener() {
                         @Override
                         public void onPositiveClick(View customView, DialogInterface
dialogInterface, int which) {
                            ViewPager viewPager = findViewById(R.id.viewPager_Fragment);
```

```
                            CommonV4Fragment commonV4Fragment =
mFragmentAdapter.getItem(viewPager.getCurrentItem());
                            if (commonV4Fragment instanceof PointCommonFragment)
                               ((PointCommonFragment)commonV4Fragment).importPointFinish();
                        }
                        @Override
                        public void onNegativeClick(View customView, DialogInterface
dialogInterface, int which) {
                        }
                    }).showDialog();
            }
               break;
            default:
               break;
         }
      }
   };
}
public class PointLibraryFragment extends PointListCommonFragment {
   private boolean mbStakeoutEnable = false;
   private boolean mbSimpleSelect = false;
   private boolean mbReturnToSurveyMain = false;
   public PointLibraryFragment(OnChangeListener listener) {
      super(listener);
   }
   public void setStakeoutReturnToMain(boolean bReturnToSurveyMain)
   {
      mbStakeoutEnable = true;
      mbReturnToSurveyMain = bReturnToSurveyMain;
   }
   public void setSimpleSelect(boolean bSimple)
   {
      mbSimpleSelect = bSimple;
   }
   @Override
   public String getTitle() {
      return AppFinalUtil.getString(R.string.title_library_coordinate_point);
   }
   protected void initCustom()
   {
      super.initCustom();
      ((CustomGridPointItemAdapter)mGridItemListAdapter).setStakeoutMode(mbStakeoutEnable);
      ((CustomGridPointItemAdapter)mGridItemListAdapter).setInvertedOrder(true);
      mGridItemListAdapter.setEnableBatchSelect(!mbReturnToSurveyMain);
      mGridItemListAdapter.setItemButtonVisible(!mbReturnToSurveyMain);
   }
   @Override
   protected void loadSearchData(ePointQueryType type, String strFind)
   {
      mAllPointListArray.clear();
```

```
mAllPointListArray.addAll(ProjectDbFileManage.getInstance().queryFindPointDataIds(type,
strFind));
    }
    @Override
    public void onClickItem(int position)
    {
        if (mbSimpleSelect)
        {
            if (position >= 0) {
                long pointId = mAllPointListArray.get(position);
                Intent intent = new Intent();
                intent.putExtra("ObjectID", pointId);
                getActivity().setResult(AppFinalUtil.RETURN_CODE_VALUE_BACK, intent);
                getActivity().finish();
            }
        }
        else {
            if (!mGridItemListAdapter.isBatchSelectMode() &&
                    !mbReturnToSurveyMain && !mbStakeoutEnable &&
                    mGridItemListAdapter.getSelectedPosition() <= 0)
            {
                onClickEdit(position);
            }
            else {
                super.onClickItem(position);
            }
        }
    }
    @Override
    public void onClickApply() {
        if (mbStakeoutEnable) {
            int selectIndex = mGridItemListAdapter.getSelectedPosition();
            if (ConfigStakeoutPoint.getInstance().isAutoStakeout()) {
                ConfigStakeoutPoint.getInstance().setAutoStakeout(false);
                ConfigStakeoutPoint.getInstance().saveConfig();
            }

            long pointId = mAllPointListArray.get(selectIndex);
            PointLibraryManage.getInstance().setStakeEntity(pointId);
            if (!mbReturnToSurveyMain) {
                Intent intent = new Intent(getActivity(), MainPointSurveyActivity.class);
                intent.putExtra(MainPointSurveyActivity.strSurveyWordModeId,
eSurveyWorkMode.WORK_MODE_STAKEOUT_POINT.swigValue());
                startActivity(intent);
            }
            getActivity().finish();
        }
    }
    @Override
    public int getHeaderLayout() {
```

```
        return R.layout.header_project_data_view;
    }
    @Override
    public ArrayList<TextView> getHeadTextViews(View convertView) {
        ArrayList<TextView> textViews = new ArrayList<>();
        TextView temTextView = convertView.findViewById(R.id.textView_point_type);
        textViews.add(temTextView);
        temTextView = convertView.findViewById(R.id.textView_Name);
        textViews.add(temTextView);
        if (ConfigSystem.getInstance().isDisplayNorthFirst()) {
            temTextView = convertView.findViewById(R.id.textView_North);
            textViews.add(temTextView);
            temTextView = convertView.findViewById(R.id.textView_East);
            textViews.add(temTextView);
        } else {
            temTextView = convertView.findViewById(R.id.textView_East);
            textViews.add(temTextView);
            temTextView = convertView.findViewById(R.id.textView_North);
            textViews.add(temTextView);
        }
        temTextView = convertView.findViewById(R.id.textView_Height);
        textViews.add(temTextView);
        temTextView = convertView.findViewById(R.id.textView_Latitude);
        textViews.add(temTextView);
        temTextView = convertView.findViewById(R.id.textView_Longitude);
        textViews.add(temTextView);
        temTextView = convertView.findViewById(R.id.textView_Altitude);
        textViews.add(temTextView);
        temTextView = convertView.findViewById(R.id.textView_Code);
        textViews.add(temTextView);
        temTextView = convertView.findViewById(R.id.textView_Mileage);
        textViews.add(temTextView);
        temTextView = convertView.findViewById(R.id.textView_Offset);
        textViews.add(temTextView);
        temTextView = convertView.findViewById(R.id.textView_BaseDistance);
        textViews.add(temTextView);
        temTextView = convertView.findViewById(R.id.textView_DateTime);
        textViews.add(temTextView);
        return textViews;
    }
    @Override
    public int getDataSize() {
        return mAllPointListArray.size();
    }
    @Override
    public ArrayList<String> getDataByIndex(int index) {
        if (index < 0 || index >= getDataSize())
            return new ArrayList<>();
        tagPointNode                                    pointNode                                    =
ProjectDbFileManage.getInstance().queryPointNode(mAllPointListArray.get(getDataSize()      -
```

```
index - 1), true);
        ArrayList<String> arrayList = new ArrayList<>();
        arrayList.add(pointNode.getPointType().swigValue() + "");
        arrayList.add(pointNode.strPointName);
        eLengthUnitType unitType = ConfigSystem.getInstance().getLengthUnitType();
        eAngleFormatType angleFormatType = ConfigSystem.getInstance().getAngleFormatType();
        tagBLHCoord blhCoord = pointNode.getBLHCoord();
        tagNEhCoord nehCoord = pointNode.getNehCoord();
        if (ConfigSystem.getInstance().isDisplayNorthFirst()) {
            arrayList.add(StringFunction.valueOf(unitType.getLength(nehCoord.getNorth())));
            arrayList.add(StringFunction.valueOf(unitType.getLength(nehCoord.getEast())));
        } else {
            arrayList.add(StringFunction.valueOf(unitType.getLength(nehCoord.getEast())));
            arrayList.add(StringFunction.valueOf(unitType.getLength(nehCoord.getNorth())));
        }
        arrayList.add(StringFunction.valueOf(unitType.getLength(nehCoord.getHeight())));
        arrayList.add(angleFormatType.toString(blhCoord.getLatitude(),
eAngleFormatType.BLMODE_B));
        arrayList.add(angleFormatType.toString(blhCoord.getLongitude(),
eAngleFormatType.BLMODE_L));
        arrayList.add(StringFunction.valueOf(unitType.getLength(blhCoord.getAltitude())));
        arrayList.add(pointNode.strPointCode);
        if (pointNode.stakeoutData != null && pointNode.stakeoutData.dataType.swigValue() >= 10)
        {
            eMileageType mileageType = ConfigSystem.getInstance().getMileageType();
            arrayList.add(mileageType.toString(pointNode.stakeoutData.dMileage));
arrayList.add(StringFunction.valueOf(unitType.getLength(pointNode.stakeoutData.dOffset)));
        }
        else {
            arrayList.add("");
            arrayList.add("");
        }
        if (pointNode.getPointType().isSurveyPoint()) {
arrayList.add(StringFunction.valueOf(unitType.getLength(pointNode.gnssPositionData.getDista
nceToBase())));
        }
        else{
            arrayList.add("");
        }
        arrayList.add(pointNode.getDateTime().toString());
        return arrayList;
    }
}
public class SurveyPointEditActivity extends CommonBaseActivity
implements OnClickListener {
    public static tagPointNode cEditPointNode = null;
    byte[] bPhotoSketch = null;
    RoverAntennaPar temAntenna = new RoverAntennaPar();
    @Override
    protected void onCreate(Bundle savedInstanceState)
```

```
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_survey_point_edit);
        if (cEditPointNode == null || cEditPointNode.gnssPositionData == null) {
            finish();
            return;
        }
        temAntenna.setType(cEditPointNode.gnssPositionData.getAntennaMeasureType());
temAntenna.setMeasureHeight(cEditPointNode.gnssPositionData.getAntennaMeasureHeight());
        temAntenna.setAntennaInfo(cEditPointNode.gnssPositionData.getAntennaInfoData());
        bPhotoSketch = ProjectDbFileManage.getInstance().getImageRemark(cEditPointNode.keyId);
        if (bPhotoSketch != null)
        {
            Button eventImage = findViewById(R.id.button_ImageNote);
            eventImage.setText(StringFunction.addRedPoint(eventImage.getText().toString()));
        }
        initialUI();
        setTextViewFocus(R.id.editText_Name);
    }
    protected void initialUI()
    {
        setTextViewText(R.id.editText_Name, cEditPointNode.strPointName);
        setTextViewText(R.id.editText_Code, cEditPointNode.strPointCode);
        initClickListener(R.id.imageButton_CodeSelect, this);
        initClickListener(R.id.button_ImageNote, this);
        initClickListener(R.id.linearLayout_Antenna, this);
        initClickListener(R.id.button_OK, this);
        eLengthUnitType unitType = ConfigSystem.getInstance().getLengthUnitType();
        String                                strAntennaHeight                                =
StringFunction.format("%s+%s%s",StringFunction.valueOf(unitType.getLength(temAntenna.getMea
sureHeight()),true),
StringFunction.valueOf(unitType.getLength(temAntenna.getAntennaHeight()-temAntenna.getMeasu
reHeight())), unitType.getUnit());
        setTextViewText(R.id.textView_AntennaValue, strAntennaHeight);
        if           (cEditPointNode.gnssPositionData.getSensorType().swigValue()           >=
eSensorType.TYPE_SENSOR_INCLINE.swigValue()) {
            setEnabled(R.id.linearLayout_Antenna, false);
        }
        displayGnssInfo();
    }
    @Override
    public void onClick(View arg0) {
        if(arg0.getId() ==R.id.button_OK)
        {
            onButtonOK();
        }else if(arg0.getId() == R.id.button_ImageNote)
        {
            Intent intent = new Intent(this, PhotoSketchActivity.class);
            PhotoSketchActivity.bPhotoSketch = bPhotoSketch;
            PhotoSketchActivity.cPointNode = cEditPointNode;
```

```
            startActivityForResult(intent, R.id.button_ImageNote);
        }
        else if (R.id.imageButton_CodeSelect == arg0.getId()) {
DisplayMenuManage.getInstance().doFunctionForId(eFunctionType.FUNCTION_TYPE_CODE_LIBRARY.sw
igValue());
        }else if (R.id.linearLayout_Antenna == arg0.getId()) {
            Intent intent = new Intent();
            intent.putExtra("AntennaMeasureType", temAntenna.getType().swigValue());
            intent.putExtra("AntennaMeasureHeight", temAntenna.getMeasureHeight());
            intent.putExtra("AntennaInfo", temAntenna.getAntennaInfo().toString());
            intent.setClass(this, SettingRoverAntennaActivity.class);
            startActivityForResult(intent, R.id.linearLayout_Antenna);
        }
    }
    private void onButtonOK() {
        boolean bChangeValue = false;
        String strPointName = getTextViewString(R.id.editText_Name);
        String strPointCode = getTextViewString(R.id.editText_Code);
        if(!strPointName.equals(cEditPointNode.strPointName) ||
              !strPointCode.equals(cEditPointNode.strPointCode))
        {
            bChangeValue = true;
            cEditPointNode.strPointName = strPointName;
            cEditPointNode.strPointCode = strPointCode;
        }
        if (cEditPointNode.gnssPositionData.getAntennaMeasureType() != temAntenna.getType() ||
              Math.abs(cEditPointNode.gnssPositionData.getAntennaMeasureHeight()
temAntenna.getMeasureHeight()) > 1E-4 ||
              !cEditPointNode.gnssPositionData.getAntennaInfoData().toString().equals(temAnte
nna.getAntennaInfo().toString()))
        {
            bChangeValue = true;
            cEditPointNode.gnssPositionData.setAntennaMeasureType(temAntenna.getType());
cEditPointNode.gnssPositionData.setAntennaMeasureHeight(temAntenna.getMeasureHeight());
            cEditPointNode.gnssPositionData.setAntennaInfoData(temAntenna.getAntennaInfo());
            cEditPointNode.gnssPositionData.updateAntennaHeight();
        }
        if (bChangeValue)
        {
            Intent intent = new Intent();
            intent.putExtra("Position", getIntent().getIntExtra("Position", -1));
            intent.putExtra("PointType", ePointType.POINT_TYPE_SURVEY.swigValue());
            setResult(AppFinalUtil.RETURN_CODE_VALUE_BACK, intent);
        }
        finish();
    }
    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event) {
        if (AppFinalUtil.isSurveyKey(keyCode)){
            onButtonOK();
```

```
            return true;
        }
        ArrayList<eFunctionType> functionTypes = new ArrayList<>();
        functionTypes.add(eFunctionType.FUNCTION_TYPE_ANTENNA_SETTING);
        functionTypes.add(eFunctionType.FUNCTION_TYPE_CODE_LIBRARY);
        eFunctionType functionType = ShortCutsManage.getInstance().getFunctionType(keyCode,
functionTypes);
        if (functionType != eFunctionType.FUNCTION_TYPE_NULL) {
            DisplayMenuManage.getInstance().doFunctionForId(functionType.swigValue());
            return true;
        }
        return super.onKeyDown(keyCode, event);
    }
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (R.id.linearLayout_Antenna == requestCode)
        {
            if (data == null)
                return;
temAntenna.setType(eAntennaMeasureType.swigToEnum(data.getIntExtra("AntennaMeasureType",
0)));
            temAntenna.setMeasureHeight(data.getDoubleExtra("AntennaMeasureHeight", 0));
            DeviceInfoAntenna infoAntenna = new DeviceInfoAntenna();
            infoAntenna.parseString(data.getStringExtra("AntennaInfo"));
            temAntenna.setAntennaInfo(infoAntenna);
            displayGnssInfo();
            eLengthUnitType unitType = ConfigSystem.getInstance().getLengthUnitType();
            String                                strAntennaHeight                                =
StringFunction.format("%s+%s%s",StringFunction.valueOf(unitType.getLength(temAntenna.getMea
sureHeight()),true),
StringFunction.valueOf(unitType.getLength(temAntenna.getAntennaHeight()-temAntenna.getMeasu
reHeight())), unitType.getUnit());
            setTextViewText(R.id.textView_AntennaValue, strAntennaHeight);
            return;
        }
        super.onActivityResult(requestCode, resultCode, data);
        if (R.id.button_ImageNote == requestCode)
        {
            if(PhotoSketchActivity.IsModified) {
                bPhotoSketch = PhotoSketchActivity.bPhotoSketch;
                Button eventImage = findViewById(R.id.button_ImageNote);
                if (bPhotoSketch != null)
                {
eventImage.setText(StringFunction.addRedPoint(getString(R.string.button_image_note)));
                }
                else {
                    eventImage.setText(getString(R.string.button_image_note));
                }
                ProjectDbFileManage.getInstance().updateImageRemark(cEditPointNode.keyId,
PhotoSketchActivity.bPhotoSketch);
```

```
            PhotoSketchActivity.bPhotoSketch = null;
         }
      }
      if(requestCode == R.id.imageButton_CodeSelect && data!=null)
      {
         String strCode = data.getStringExtra("resultValue");
         if (strCode != null)
         {
            setTextViewText(R.id.editText_Code, strCode);
         }
      }
   }
   public void displayGnssInfo() {
      GnssRecordPositionData temGnssData = cEditPointNode.gnssPositionData;
      double dAddAntennaHeight = temAntenna.getAntennaHeight() - temGnssData.getPhaseHeight();
      String strTemString = "";
      strTemString = StringFunction.format("%s/%d", temGnssData.getSatInLock(),
temGnssData.getSatInView());
      setTextViewText(R.id.editText_SolutionState, temGnssData.getStatusMark());
      if (strTemString.length() > 0)
         setTextViewText(R.id.editText_SatelliteNum, String.format("(%s)", strTemString));
      else
         setTextViewText(R.id.editText_SatelliteNum, strTemString);
      eLengthUnitType unitType = ConfigSystem.getInstance().getLengthUnitType();
      eAngleFormatType angleFormatType = ConfigSystem.getInstance().getAngleFormatType();
      CustomTextViewListLayout linearLayoutCoordinate=
findViewById(R.id.linearLayout_Coordinate);
      linearLayoutCoordinate.clear();
      tagBLHCoord blhCoord = temGnssData.getWgs84BlhCoord();
      tagNEhCoord temNehCoord = temGnssData.getNehCoord();
      strTemString = angleFormatType.toString(blhCoord.getLatitude(),
eAngleFormatType.BLMODE_B);
      if (ConfigSystem.getInstance().isDisplayNorthFirst()) {
linearLayoutCoordinate.add(getString(R.string.string_lat),strTemString,getString(R.string.s
tring_n),StringFunction.valueOf(unitType.getLength(temNehCoord.getNorth())));
      }
      else {
linearLayoutCoordinate.add(getString(R.string.string_lat),strTemString,getString(R.string.s
tring_e),StringFunction.valueOf(unitType.getLength(temNehCoord.getEast())));
      }
      strTemString = angleFormatType.toString(blhCoord.getLongitude(),
eAngleFormatType.BLMODE_L);
      if (ConfigSystem.getInstance().isDisplayNorthFirst()) {
linearLayoutCoordinate.add(getString(R.string.string_lon),strTemString,getString(R.string.s
tring_e),StringFunction.valueOf(unitType.getLength(temNehCoord.getEast())));
      }
      else {
linearLayoutCoordinate.add(getString(R.string.string_lon),strTemString,getString(R.string.s
tring_n),StringFunction.valueOf(unitType.getLength(temNehCoord.getNorth())));
      }
```

```
linearLayoutCoordinate.add(getString(R.string.string_alt),StringFunction.valueOf(unitType.getLength(blhCoord.getAltitude()-dAddAntennaHeight)),getString(R.string.string_h),StringFunction.valueOf(unitType.getLength(temNehCoord.getHeight()-dAddAntennaHeight)));
      if (cEditPointNode.gnssPositionData != null) {
         boolean bVaild = true;
         CCoordinateConvert coordinateConvert = new CCoordinateConvert();
         tagRtcmCoordinateSystemParameter rtcmCoordinateSystemParameter = cEditPointNode.gnssPositionData.getRtcmCoordinateSystemParameter();
         if (rtcmCoordinateSystemParameter != null)
         {
coordinateConvert.setDestEllipsoidParameter(rtcmCoordinateSystemParameter.getTargetEllipsoidParam());
coordinateConvert.setProjectionParameter(rtcmCoordinateSystemParameter.getProjectParam());
         }
         else {
            tagCoordinateSystemParameter systemParameter = cEditPointNode.gnssPositionData.getCoordinateSystemParameter();
            if (systemParameter != null) {

coordinateConvert.setDestEllipsoidParameter(systemParameter.getEllipsoidParam());
               coordinateConvert.setProjectionParameter(systemParameter.getProjectParam());
            }
            else {
               bVaild = false;
            }
         }
         if (bVaild)
         {
            double dDistance = CommonFunction.GetDistance(blhCoord.getLatitude() - 1E-6, blhCoord.getLongitude() - 1E-6, blhCoord.getAltitude(),
                  blhCoord.getLatitude() + 1E-6, blhCoord.getLongitude() + 1E-6, blhCoord.getAltitude());
            tagBLHCoord temBlhCoord = new tagBLHCoord();
            temBlhCoord.setLatitude(blhCoord.getLatitude() - 1E-6);
            temBlhCoord.setLongitude(blhCoord.getLongitude() - 1E-6);
            temBlhCoord.setAltitude(blhCoord.getAltitude());
            tagNEhCoord tagNEhCoord1 = new tagNEhCoord();
            coordinateConvert.BLHtoNEh(temBlhCoord, tagNEhCoord1);
            temBlhCoord.setLatitude(blhCoord.getLatitude() + 1E-6);
            temBlhCoord.setLongitude(blhCoord.getLongitude() + 1E-6);
            tagNEhCoord tagNEhCoord2 = new tagNEhCoord();
            coordinateConvert.BLHtoNEh(temBlhCoord, tagNEhCoord2);
            double K = CommonFunction.GetDistance(tagNEhCoord1, tagNEhCoord2) / dDistance;
linearLayoutCoordinate.add(getString(R.string.label_scale_correction_factor),StringFunction.valueOf(K, 10));
         }
      }
      BaseChangeCorrectParameter correctParameter = temGnssData.getBaseCorrectParameter();
      if (correctParameter != null)
      {
```

```
            tagNEhCoord nehCoord = correctParameter.getCorrectNehCoordParameter();
            if (ConfigSystem.getInstance().isDisplayNorthFirst()) {
               linearLayoutCoordinate.add(getString(R.string.label_adjustment_value_north),
StringFunction.valueOf(unitType.getLength(nehCoord.getNorth())));
               linearLayoutCoordinate.add(getString(R.string.label_adjustment_value_east),
StringFunction.valueOf(unitType.getLength(nehCoord.getEast())));
            }
            else
            {
               linearLayoutCoordinate.add(getString(R.string.label_adjustment_value_east),
StringFunction.valueOf(unitType.getLength(nehCoord.getEast())));
               linearLayoutCoordinate.add(getString(R.string.label_adjustment_value_north),
StringFunction.valueOf(unitType.getLength(nehCoord.getNorth())));
            }
            linearLayoutCoordinate.add(getString(R.string.label_adjustment_value_height),
StringFunction.valueOf(unitType.getLength(nehCoord.getHeight())));
         }
         CustomTextViewListLayout                                    linearLayoutPrecision=
findViewById(R.id.linearLayout_Precision);
         linearLayoutPrecision.clear();
         strTemString = StringFunction.valueOf(temGnssData.getVelocity() * 3.6);
         linearLayoutPrecision.add(getString(R.string.label_point_detail_speed),
StringFunction.valueOf(Math.ceil(temGnssData.getAgeOfDiff()),                    true),
getString(R.string.label_point_detail_heading),strTemString);
         strTemString = "NA";
         if(temGnssData.getPdop() > 1E-4) {
            strTemString = StringFunction.valueOf(temGnssData.getPdop());
         }
         String strTemString2 = "NA";
         if(temGnssData.getHrms() > 1E-4) {
            strTemString2 = StringFunction.valueOf(unitType.getLength(temGnssData.getHrms()));
         }
         linearLayoutPrecision.add(getString(R.string.label_point_detail_pdop),   strTemString,
getString(R.string.label_point_detail_hrms),strTemString2);
         strTemString = "NA";
         if(temGnssData.getHdop() > 1E-4) {
            strTemString = StringFunction.valueOf(temGnssData.getHdop());
         }
         strTemString2 = "NA";
         if(temGnssData.getVrms() > 1E-4) {
            strTemString2 = StringFunction.valueOf(unitType.getLength(temGnssData.getVrms()));
         }
         linearLayoutPrecision.add(getString(R.string.label_point_detail_hdop),   strTemString,
getString(R.string.label_point_detail_vrms),strTemString2);
         strTemString = "NA";
         if(temGnssData.getVrms() > 1E-4) {
            strTemString = StringFunction.valueOf(temGnssData.getVdop());
         }
         linearLayoutPrecision.add(getString(R.string.label_point_detail_vdop),   strTemString,
getString(R.string.label_point_detail_diff_delay),StringFunction.valueOf(Math.ceil(temGnssD
```

```
ata.getAgeOfDiff()), true));
      linearLayoutPrecision.add(getString(R.string.string_record_average_count),
String.valueOf(temGnssData.getEpochCount()),
getString(R.string.string_record_average_count),String.valueOf(temGnssData.getElevMask()));
      CustomTextViewListLayout linearLayoutOther= findViewById(R.id.linearLayout_Other);
      linearLayoutOther.clear();
      tagGnssRefStationItem refStationItem = temGnssData.getRefStationData();
      if (refStationItem != null) {
         linearLayoutOther.add(getString(R.string.label_point_detail_base_id),
refStationItem.getBaseId(),
getString(R.string.label_point_detail_ref_distance),StringFunction.valueOf(unitType.getLeng
th(temGnssData.getDistanceToBase())));
      }
      linearLayoutOther.add(getString(R.string.label_point_detail_utcTime),
temGnssData.getDateTime().toString());
      linearLayoutOther.add(getString(R.string.label_point_detail_localTime),
temGnssData.getLocalDateTime().toString());
      linearLayoutOther.add(getString(R.string.string_instrument_serial_no),
temGnssData.getDeviceSN());
   }
}
public class SurveySettingActivity extends CommonBaseFragmentActivity implements
OnClickListener {
   private SettingFragmentAdapter mFragmentAdapter;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_survey_setting);
      initialUI();
   }
   private void initialUI() {
      initClickListener(R.id.button_OK, this);
      initClickListener(R.id.button_Default, this);
      initClickListener(R.id.button_Backspace, this);
      initClickListener(R.id.button_Clear, this);
      eSurveyWorkMode surveyWorkMode =
eSurveyWorkMode.swigToEnum(getIntent().getIntExtra("SurveyWorkMode",
WordModeManage.getInstance().getWorkMode().swigValue()));
      ePointRecordMode recordMode = ePointRecordMode.POINT_RECORD_MODE_SMOOTH;
      if (surveyWorkMode == eSurveyWorkMode.WORK_MODE_SURVEY_CONTROL_POINT)
      {
         recordMode = ePointRecordMode.POINT_RECORD_MODE_CONTROL;
      }
      else if (surveyWorkMode == eSurveyWorkMode.WORK_MODE_SURVEY_AUTO_POINT)
      {
         recordMode = ePointRecordMode.POINT_RECORD_MODE_CONTINUUM;
      }
      else {
         if (RecordPointBase.getInstance().getPointRecordMode() ==
ePointRecordMode.POINT_RECORD_MODE_BASE_SURVEY)
```

```
            recordMode = ePointRecordMode.POINT_RECORD_MODE_BASE_SURVEY;
        else if (ConfigPointSurvey.getInstance().isEnablePPK())
            recordMode = ePointRecordMode.POINT_RECORD_MODE_STOP_GO;
    }
    mFragmentAdapter = new SettingFragmentAdapter(getSupportFragmentManager(), this);
    Bundle args=new Bundle();
    args.putInt("mode", recordMode.swigValue());
    if (surveyWorkMode == eSurveyWorkMode.WORK_MODE_CAD_EDIT)
    {
        mFragmentAdapter.addFragment(new SettingCadEditFragment());
        mFragmentAdapter.addFragment(new SettingMenuDisplayFragment());
    }
    else {
        SettingRecordItemConfigFragment fragment = new SettingRecordItemConfigFragment();
        fragment.setArguments(args);
        mFragmentAdapter.addFragment(fragment);
        switch (surveyWorkMode) {
            case WORK_MODE_STAKEOUT_POINT:
                mFragmentAdapter.addFragment(new SettingPointStakeoutFragment());
                break;
            case WORK_MODE_STAKEOUT_OBJECT:
                mFragmentAdapter.addFragment(new SettingObjectStakeoutFragment());
                break;
        }
        if (surveyWorkMode != eSurveyWorkMode.WORK_MODE_SURVEY_TEXT &&
                surveyWorkMode != eSurveyWorkMode.WORK_MODE_SURVEY_CONTROL_POINT &&
                surveyWorkMode != eSurveyWorkMode.WORK_MODE_SURVEY_AUTO_POINT) {
            mFragmentAdapter.addFragment(new SettingInfoDisplayFragment());
            mFragmentAdapter.addFragment(new SettingMenuDisplayFragment());
        }
    }
    ViewPager viewPager = findViewById(R.id.viewPager_Fragment);
    viewPager.setAdapter(mFragmentAdapter);
    viewPager.addOnPageChangeListener(new ViewPager.OnPageChangeListener() {
        @Override
        public void onPageScrolled(int i, float v, int i1) {
        }
        @Override
        public void onPageSelected(int i) {
            ViewPager viewPager = findViewById(R.id.viewPager_Fragment);
            if (mFragmentAdapter.getItem(viewPager.getCurrentItem()).isSupportClear())
            {
                setVisibility(R.id.button_Clear, View.VISIBLE);
            }
            else {
                setVisibility(R.id.button_Clear, View.GONE);
            }
            if (mFragmentAdapter.getItem(viewPager.getCurrentItem()).isSupportBackspace())
            {
                setVisibility(R.id.button_Backspace, View.VISIBLE);
```

```
            }
            else {
                setVisibility(R.id.button_Backspace, View.GONE);
            }
        }
        @Override
        public void onPageScrollStateChanged(int i) {
        }
    });
    TabLayout tabLayout = findViewById(R.id.tabLayout_Fragment);
    tabLayout.setupWithViewPager(viewPager);
    if(mFragmentAdapter.getCount() <= 1)
        tabLayout.setVisibility(View.GONE);
}
@Override
public void onClick(View arg0) {
    switch (arg0.getId()) {
        case R.id.button_Default:
            OnUseDefaultConfig();
            break;
        case R.id.button_Clear:
            OnClearConfig();
            break;
        case R.id.button_Backspace: {
            ViewPager viewPager = findViewById(R.id.viewPager_Fragment);
            mFragmentAdapter.getItem(viewPager.getCurrentItem()).onButtonBackspace();
        }
            break;
        case R.id.button_OK: {
            for (int i=0; i<mFragmentAdapter.getCount(); i++) {
                mFragmentAdapter.getItem(i).updateValueFormUI();
            }
            setResult(AppFinalUtil.RETURN_CODE_DEFAULT);
            finish();
        }
        break;
        default:
            break;
    }
}
private void OnUseDefaultConfig() {
    new
CustomDialog(this,R.string.string_prompt,R.string.string_prompt_message_used_default_settin
g,R.string.button_ok,R.string.button_cancel)
            .setDialogListener(new CustomDialog.OnDialogListener() {
                @Override
                public void onPositiveClick(View customView, DialogInterface dialogInterface,
int which) {
                    ViewPager viewPager = findViewById(R.id.viewPager_Fragment);
                    mFragmentAdapter.getItem(viewPager.getCurrentItem()).onButtonDefault();
```

```
                }
                @Override
                public void onNegativeClick(View customView, DialogInterface dialogInterface,
int which) {
                }
            }).showDialog();
    }
    private void OnClearConfig() {
        new
CustomDialog(this,R.string.string_prompt,R.string.string_prompt_message_delete_setting
            ,R.string.button_ok,R.string.button_cancel)
            .setDialogListener(new CustomDialog.OnDialogListener() {
                @Override
                public void onPositiveClick(View customView, DialogInterface dialogInterface,
int which) {
                    ViewPager viewPager = findViewById(R.id.viewPager_Fragment);
                    mFragmentAdapter.getItem(viewPager.getCurrentItem()).onButtonClear();
                }
                @Override
                public void onNegativeClick(View customView, DialogInterface dialogInterface,
int which) {
                }
            }).showDialog();
    }
    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        ViewPager viewPager = findViewById(R.id.viewPager_Fragment);
mFragmentAdapter.getItem(viewPager.getCurrentItem()).onActivityResult(requestCode&0xFFFF,
resultCode, data);
    }
}
public class SettingInfoDisplayFragment extends SettingFragment {
    ArrayList<Integer> mSelectArrayList = new ArrayList<Integer>();
    private DisplayConfig mDisplayConfig = null;
    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, Bundle
savedInstanceState) {
        if (mInfoView != null)
            return mInfoView;
        mInfoView                                                                    =
inflater.inflate(R.layout.layout_fragment_display_info_select,container,false);
        eSurveyWorkMode wordMode = WordModeManage.getInstance().getWorkMode();
        mDisplayConfig = ConfigDisplayInfoManage.getInstance().getDisplayConfig(wordMode);
        mSelectArrayList = mDisplayConfig.getSelectDisplayList();
        CustomItemListLayout                          itemListLayout                 =
mInfoView.findViewById(R.id.linearLayout_ItemList);
        itemListLayout.setColumnCount(AppFinalUtil.isLanguageZh()?3:2);
        itemListLayout.setOnClickListener(new CustomListItemRow.OnRowClickListener() {
            @Override
```

```
            public void onClick(String strValue, int id) {
                if(mSelectArrayList.size() >= 8) {
                    return;
                }
                eDisplayItemType fieldType = eDisplayItemType.swigToEnum(id);
                mSelectArrayList.add(fieldType.swigValue());
                updateValueToUI();
            }
            @Override
            public void onClickDelete(String strValue, int id) {
            }
        });
        updateValueToUI();
        setCheckButtonChecked(R.id.checkButton_DisplayName,
ConfigSystem.getInstance().isDisplayMapName());
setCheckButtonChecked(R.id.checkButton_DisplayCode,ConfigSystem.getInstance().isDisplayMapC
ode());
setCheckButtonChecked(R.id.checkButton_DisplayHeight,ConfigSystem.getInstance().isDisplayMa
pHeight());
        CustomTextViewLayoutSelect                          temLayoutSelect                          =
mInfoView.findViewById(R.id.linearLayout_BackgroundColor);
        temLayoutSelect.clearItem();
        temLayoutSelect.addItem(getString(R.string.string_color_white));
        temLayoutSelect.addItem(getString(R.string.string_color_black));
        temLayoutSelect.setSelectedId(ConfigSystem.getInstance().getBackgroundColor()       ==
Color.BLACK ? 1:0);
        return mInfoView;
    }
    @Override
    public void onButtonDefault()
    {
        mSelectArrayList = mDisplayConfig.getDefaultDisPlayItems();
        updateValueToUI();
    }
    public boolean isSupportBackspace()
    {
        return true;
    }
    @Override
    public void onButtonBackspace()
    {
        if (mSelectArrayList.size() > 0)
            mSelectArrayList.remove(mSelectArrayList.size()-1);
        updateValueToUI();
    }
    @Override
    public String getTitle() {
        return AppFinalUtil.getString(R.string.string_display_info);
    }
    @Override
```

```
    public void updateValueToUI() {
        if (mInfoView == null)
            return;
        CustomInfoView infoView = mInfoView.findViewById(R.id.custom_display_info_view);
        infoView.requestLayout(mSelectArrayList);
        infoView.invalidate();
        CustomItemListLayout itemListLayout =
mInfoView.findViewById(R.id.linearLayout_ItemList);
        itemListLayout.clear();
        ArrayList<Integer> enableSelectList =
mDisplayConfig.getDisplayItemsNoInList(mSelectArrayList);
        for(int i=0;i<enableSelectList.size();i++){
            eDisplayItemType fieldType = eDisplayItemType.swigToEnum(enableSelectList.get(i));
            itemListLayout.add(fieldType.getTitle(), fieldType.swigValue());
        }
        itemListLayout.updateValue();
    }
    @Override
    public void updateValueFormUI() {
        if (mInfoView == null)
            return;
        if (mDisplayConfig != null) {
            mDisplayConfig.setSelectDisplayList(mSelectArrayList);
            ConfigDisplayInfoManage.getInstance().saveConfig();
        }
ConfigSystem.getInstance().setDisplayMapName(getCheckButtonChecked(R.id.checkButton_Display
Name));
ConfigSystem.getInstance().setDisplayMapCode(getCheckButtonChecked(R.id.checkButton_Display
Code));
ConfigSystem.getInstance().setDisplayMapHeight(getCheckButtonChecked(R.id.checkButton_Displ
ayHeight));
        CustomTextViewLayoutSelect temLayoutSelect =
mInfoView.findViewById(R.id.linearLayout_BackgroundColor);
        ConfigSystem.getInstance().setBackgroundColor(temLayoutSelect.getSelectedId() ==
1?Color.BLACK:Color.WHITE);
        ConfigSystem.getInstance().saveConfig();
    }
}
public class SettingPointStakeoutFragment extends SettingFragment{
    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, Bundle
savedInstanceState) {
        if (mInfoView != null)
            return mInfoView;
        mInfoView = inflater.inflate(R.layout.layout_setting_stakeout_point, container,false);
        eLengthUnitType unitType = ConfigSystem.getInstance().getLengthUnitType();
        setCheckButtonChecked(R.id.checkButton_Stakeout_Direction,
ConfigStakeout.getInstance().isPromptStakeoutDirection());
        CustomTextViewLayoutSelect layoutSelect =
```

```
mInfoView.findViewById(R.id.layoutSelect_Direction_Reference);
      layoutSelect.clearItem();
      ArrayList<eDirectionReference>                directionReferences                =
DeviceCommand.getInstance().getAvailableDirectionReferenceTypes();
      for (int i=0; i< directionReferences.size(); i++){
         eDirectionReference directionReference = directionReferences.get(i);
         layoutSelect.addItem(directionReference.getTitle(), directionReference.swigValue());
      }
      layoutSelect.registerItemSelectedInterface(new
CustomTextViewLayoutSelect.OnItemSelectedCallback() {
         @Override
         public void onItemSelected(View view, String label, int id) {
            int       visibility       =       eDirectionReference.swigToEnum(id)       ==
eDirectionReference.TYPE_REFERENCE_POINT?View.VISIBLE:View.GONE;
            setVisibility(R.id.viewListLayout_ReferencePoint, visibility);
            setVisibility(R.id.view_line_ReferencePoint, visibility);
         }
      });
layoutSelect.setSelectedId(ConfigStakeout.getInstance().getStakeoutReference().swigValue());
      EntityPoint referencePoint = ConfigStakeout.getInstance().getReferencePoint();
      CustomTextViewListLayout                viewListLayoutReferencePoint                =
mInfoView.findViewById(R.id.viewListLayout_ReferencePoint);
      viewListLayoutReferencePoint.setName(referencePoint.strName);
      viewListLayoutReferencePoint.clear();
      if (ConfigSystem.getInstance().isDisplayNorthFirst())
         viewListLayoutReferencePoint.add(StringFunction.format("%s:%s",
AppFinalUtil.getString(R.string.string_display_bar_north),
StringFunction.valueOf(unitType.getLength(referencePoint.dNorth))),                "",
StringFunction.format("%s:%s",     AppFinalUtil.getString(R.string.string_display_bar_east),
StringFunction.valueOf(unitType.getLength(referencePoint.dEast))), "");
      else
         viewListLayoutReferencePoint.add(StringFunction.format("%s:%s",
AppFinalUtil.getString(R.string.string_display_bar_east),
StringFunction.valueOf(unitType.getLength(referencePoint.dEast))),                "",
StringFunction.format("%s:%s",    AppFinalUtil.getString(R.string.string_display_bar_north),
StringFunction.valueOf(unitType.getLength(referencePoint.dNorth))), "");
      viewListLayoutReferencePoint.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            EntityPoint referencePoint = ConfigStakeout.getInstance().getReferencePoint();
            Intent intent = new Intent();
            intent.putExtra("VaildCoordType", 0x01);
            intent.putExtra("CoordinateType", eCoordinateType.TYPE_COORD_NEH.swigValue());
            intent.putExtra("InputName", true);
            intent.putExtra("Name", referencePoint.strName);
            intent.putExtra("North", referencePoint.dNorth);
            intent.putExtra("East", referencePoint.dEast);
            intent.putExtra("Height", referencePoint.dHeight);
            intent.setClass(getContext(), CoordinateInputActivity.class);
            startActivityForResult(intent, R.id.viewListLayout_ReferencePoint&0xFFFF);
```

```
            }
        });
        CustomTextViewLayoutSelectEdit layoutSelectEdit =
mInfoView.findViewById(R.id.linearLayout_PromptRange);
        String strTitle = StringFunction.format("%s(%s)",
getString(R.string.string_stakeout_prompt_range), unitType.getUnit());
        layoutSelectEdit.setTitle(strTitle);
        layoutSelectEdit.addItems(new String[]{"0.5", "1","1.5","2","3","5","10"});
        setUnitDoubleValue(R.id.linearLayout_PromptRange,
ConfigStakeout.getInstance().getPromptRange());
        layoutSelectEdit = mInfoView.findViewById(R.id.linearLayout_RecordRange);
        strTitle = StringFunction.format("%s(%s)",
getString(R.string.string_stakeout_record_range), unitType.getUnit());
        layoutSelectEdit.setTitle(strTitle);
        layoutSelectEdit.addItems(new String[]{"0.02","0.05","0.1","0.2","0.5","1","10"});
        setUnitDoubleValue(R.id.linearLayout_RecordRange,
ConfigStakeout.getInstance().getStakeoutRecordRange());
        CustomCheckButton checkButton =
mInfoView.findViewById(R.id.checkButton_AutoStakeCompass);
        checkButton.setChecked(ConfigStakeout.getInstance().isAutoEnterStakeCompass());
        checkButton = mInfoView.findViewById(R.id.checkButton_AutoZoom);
        checkButton.setChecked(ConfigStakeout.getInstance().isAutoZoom());
        checkButton = mInfoView.findViewById(R.id.checkButton_AutoStakeout);
        checkButton.setChecked(ConfigStakeoutPoint.getInstance().isAutoStakeout());
        checkButton = mInfoView.findViewById(R.id.checkButton_AutoMarkStakeout);
        checkButton.setChecked(ConfigStakeoutPoint.getInstance().isAutoMarkStakeout());
        checkButton.setVisibility(View.GONE);
        return mInfoView;
    }
    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if(resultCode == AppFinalUtil.RETURN_CODE_VALUE_BACK) {
            if ((requestCode&0xFFFF) == (R.id.viewListLayout_ReferencePoint&0xFFFF)) {
                EntityPoint referencePoint = ConfigStakeout.getInstance().getReferencePoint();
                referencePoint.strName = data.getStringExtra("Name");
                referencePoint.dNorth = data.getDoubleExtra("North", 0);
                referencePoint.dEast = data.getDoubleExtra("East", 0);
                referencePoint.dHeight = data.getDoubleExtra("Height", 0);
                CustomTextViewListLayout viewListLayoutReferencePoint =
mInfoView.findViewById(R.id.viewListLayout_ReferencePoint);
                viewListLayoutReferencePoint.setName(referencePoint.strName);
                viewListLayoutReferencePoint.clear();
                eLengthUnitType unitType = ConfigSystem.getInstance().getLengthUnitType();
                if (ConfigSystem.getInstance().isDisplayNorthFirst())
                    viewListLayoutReferencePoint.add(StringFunction.format("%s:%s",
AppFinalUtil.getString(R.string.string_display_bar_north),
StringFunction.valueOf(unitType.getLength(referencePoint.dNorth))), "",
StringFunction.format("%s:%s",     AppFinalUtil.getString(R.string.string_display_bar_east),
StringFunction.valueOf(unitType.getLength(referencePoint.dEast))), "");
```

```
            else
                viewListLayoutReferencePoint.add(StringFunction.format("%s:%s",
AppFinalUtil.getString(R.string.string_display_bar_east),
StringFunction.valueOf(unitType.getLength(referencePoint.dEast))),                    "",
StringFunction.format("%s:%s",    AppFinalUtil.getString(R.string.string_display_bar_north),
StringFunction.valueOf(unitType.getLength(referencePoint.dNorth))), "");
          }
       }
    }
    @Override
    public void onButtonDefault() {
       CustomCheckButton                          checkButton                          =
mInfoView.findViewById(R.id.checkButton_AutoStakeCompass);
       checkButton.setChecked(false);
       checkButton = mInfoView.findViewById(R.id.checkButton_AutoZoom);
       checkButton.setChecked(false);
       checkButton = mInfoView.findViewById(R.id.checkButton_AutoStakeout);
       checkButton.setChecked(false);
       checkButton = mInfoView.findViewById(R.id.checkButton_AutoMarkStakeout);
       checkButton.setChecked(false);
       eLengthUnitType unitType = ConfigSystem.getInstance().getLengthUnitType();
       checkButton = mInfoView.findViewById(R.id.checkButton_Stakeout_Direction);
       checkButton.setChecked(DeviceCommand.getInstance().isTotalStation());
       CustomTextViewLayoutSelect                    layoutSelect                    =
mInfoView.findViewById(R.id.layoutSelect_Direction_Reference);
       layoutSelect.setSelectedId(-1);
       CustomTextViewLayoutSelectEdit                    layoutSelectEdit                    =
mInfoView.findViewById(R.id.linearLayout_PromptRange);
       layoutSelectEdit.setText(StringFunction.valueOf(unitType.getLength(1), true));
       layoutSelectEdit = mInfoView.findViewById(R.id.linearLayout_RecordRange);
       layoutSelectEdit.setText(StringFunction.valueOf(unitType.getLength(0.02), true));
    }
    @Override
    public String getTitle() {
       return AppFinalUtil.getString(R.string.title_stakeout_setting);
    }
    @Override
    public void updateValueToUI() {
    }
    @Override
    public void updateValueFormUI() {
       if (mInfoView == null)
          return;
       CustomCheckButton                          checkButton                          =
mInfoView.findViewById(R.id.checkButton_AutoStakeCompass);
       ConfigStakeout.getInstance().setAutoEnterStakeCompass(checkButton.isChecked());
       checkButton = mInfoView.findViewById(R.id.checkButton_AutoZoom);
       ConfigStakeout.getInstance().setAutoZoom(checkButton.isChecked());
       checkButton = mInfoView.findViewById(R.id.checkButton_AutoStakeout);
       ConfigStakeoutPoint.getInstance().setAutoStakeout(checkButton.isChecked());
```

```
        checkButton = mInfoView.findViewById(R.id.checkButton_AutoMarkStakeout);
        ConfigStakeoutPoint.getInstance().setAutoMarkStakeout(checkButton.isChecked());
        eLengthUnitType unitType = ConfigSystem.getInstance().getLengthUnitType();
        CustomTextViewLayoutSelectEdit layoutSelectEdit =
mInfoView.findViewById(R.id.linearLayout_PromptRange);
ConfigStakeout.getInstance().setPromptRange(unitType.getMeterLength(layoutSelectEdit.getDou
bleValue()));
        layoutSelectEdit = mInfoView.findViewById(R.id.linearLayout_RecordRange);
ConfigStakeout.getInstance().setStakeoutRecordRange(unitType.getMeterLength(layoutSelectEdi
t.getDoubleValue()));
        checkButton = mInfoView.findViewById(R.id.checkButton_Stakeout_Direction);
        ConfigStakeout.getInstance().setPromptStakeoutDirection(checkButton.isChecked());
        CustomTextViewLayoutSelect layoutSelect =
mInfoView.findViewById(R.id.layoutSelect_Direction_Reference);
ConfigStakeout.getInstance().setStakeoutReference(eDirectionReference.swigToEnum(layoutSele
ct.getSelectedId()));
        ConfigStakeoutPoint.getInstance().saveConfig();
        ConfigStakeout.getInstance().saveConfig();
    }
}
public class SettingMenuDisplayFragment extends SettingFragment{
    private DisplayConfig mDisplayConfig = null;
    private ArrayList<Integer> mSelectArrayList = new ArrayList<Integer>();
    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, Bundle
savedInstanceState) {
        if (mInfoView != null)
            return mInfoView;
        mInfoView =
inflater.inflate(R.layout.layout_fragment_menu_info_select,container,false);
        eSurveyWorkMode wordMode = WordModeManage.getInstance().getWorkMode();
        mDisplayConfig = ConfigDisplayInfoManage.getInstance().getDisplayConfig(wordMode);
        mSelectArrayList = mDisplayConfig.getSelectMenuList();
        CustomItemListLayout itemListLayout =
mInfoView.findViewById(R.id.linearLayout_SelectList);
        itemListLayout.setColumnCount(1);
        itemListLayout.setOnClickListener(new CustomListItemRow.OnRowClickListener() {
            @Override
            public void onClick(String strValue, int id) {
                eFunctionType functionType = eFunctionType.swigToEnum(id);
                mSelectArrayList.add(functionType.swigValue());
                updateValueToUI();
            }
            @Override
            public void onClickDelete(String strValue, int id) {
            }
        });
        itemListLayout = mInfoView.findViewById(R.id.linearLayout_SelectedList);
        itemListLayout.setColumnCount(1);
```

```
        itemListLayout.setOnClickListener(new CustomListItemRow.OnRowClickListener() {
            @Override
            public void onClick(String strValue, int id) {
                eFunctionType functionType = eFunctionType.swigToEnum(id);
                mSelectArrayList.remove(Integer.valueOf(functionType.swigValue()));
                updateValueToUI();
            }
            @Override
            public void onClickDelete(String strValue, int id) {
            }
        });
        return mInfoView;
    }
    @Override
    public void onButtonDefault()
    {
        mSelectArrayList = mDisplayConfig.getDefaultMenuItems();
        updateValueToUI();
    }
    public boolean isSupportClear()
    {
        return true;
    }
    @Override
    public void onButtonClear()
    {
        mSelectArrayList.clear();
        updateValueToUI();
    }
    @Override
    public String getTitle() {
        return AppFinalUtil.getString(R.string.string_display_menu);
    }
    @Override
    public void updateValueToUI() {
        if (mInfoView == null)
            return;
        CustomItemListLayout itemListLayout =
mInfoView.findViewById(R.id.linearLayout_SelectList);
        itemListLayout.clear();
        ArrayList<Integer> listAvailable =
mDisplayConfig.getMenuItemsNoInList(mSelectArrayList);
        for(Integer id:listAvailable){
            if (id == eFunctionType.FUNCTION_TYPE_ELECTRON_BUBBLE.swigValue() &&
                DeviceCommandParse.getInstance().getSensorModel() != ESensorModel.TiltSurvey)
                continue;
            itemListLayout.add(eFunctionType.swigToEnum(id));
        }
        itemListLayout.updateValue();
        itemListLayout = mInfoView.findViewById(R.id.linearLayout_SelectedList);
```

```
        itemListLayout.clear();
        for(int i = mSelectArrayList.size()-1; i>=0; i--){
            itemListLayout.add(eFunctionType.swigToEnum(mSelectArrayList.get(i)));
        }
        itemListLayout.updateValue();
    }
    @Override
    public void updateValueFormUI() {
        if (mDisplayConfig != null) {
            mDisplayConfig.setSelectMenuList(mSelectArrayList);
            ConfigDisplayInfoManage.getInstance().saveConfig();
        }
    }
}
public class AboutProgramActivity extends CommonEventBaseActivity implements OnClickListener
{
    private RegisterTransOutSocketClient transOutSocketClient = new
RegisterTransOutSocketClient();
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_program_about);
        if (AppFinalUtil.getAppID() == eAppIdentifierType.APP_ID_SURVEY_GINTEC)
            setVisibility(R.id.imageView_Icon, View.VISIBLE);
        setVisibility(R.id.button_Copy, View.GONE);
        initialUI();
        return;
    }
    private void initialUI()
    {
        String strVersion = StringFunction.format("V%s", AppFinalUtil.getAppVersion());
        initClickListener(R.id.button_New_Version, this);
        initClickListener(R.id.button_Activation, this);
        initClickListener(R.id.button_Activation_Trans_out, this);
        initClickListener(R.id.button_Feedback, this);
        initClickListener(R.id.button_Apply, this);
        initClickListener(R.id.textViewDeviceId, this);
        String strCompany = AppFinalUtil.getAppID().getCompanyName();
        ImageView temTextView = findViewById(R.id.imageView_logo);
        try {
            String strImageFileName = "";
            strImageFileName = StringFunction.format("%s/%s",
ProjectManage.GetInstance().getProgramCachePath(), ".app_qr_code.png");
            FileInputStream fis = new FileInputStream(strImageFileName);
            temTextView.setImageBitmap(BitmapFactory.decodeStream(fis));
        } catch (Exception e) {
            temTextView.setImageResource(AppFinalUtil.getAppID().getQRCodeId());
        }
        setTextViewText(R.id.textView_Version, strVersion);
```

```
        setTextViewText(R.id.textView_Company, strCompany);
        setTextViewText(R.id.textView_Copyright, AppFinalUtil.getAppID().getCopyright());
        if(VersionDownloadClient.isUpdateAvailable()){
           Button eventAd = findViewById(R.id.button_New_Version);
           setTextViewText(R.id.button_New_Version, getString(R.string.label_new_version));
           eventAd.setText(StringFunction.addRedPoint(eventAd.getText().toString()));
        }
        updateActivationInfo();
    }
    @Override
    public void finish()
    {
        transOutSocketClient.disConnect();
        super.finish();
    }
    private void updateActivationInfo() {
        int                                overDateFunc                                =
RegisterManage.GetInstance().isRegOverDate(eMainMenuType.MENU_TYPE_NULL);
        boolean displayPVersion = false;
        if (AppFinalUtil.getAppID().isChinaVersion() &&
              (RegisterManage.GetInstance().getValidFunction()&0x02) <= 0)
        {
           if (AppFinalUtil.isLanguageZhCh())
           {
                 if (CustomGnssData.getInstance().isValidate() &&
                   CustomGnssData.getInstance().getSolutionType()                    !=
eGnssSolutionType.FIX_TYPE_INVALID)
                 {
                    boolean                          bPointInChina                          =
RegisterManage.GetInstance().pointInChina(CustomGnssData.getInstance().getLatitude(),
CustomGnssData.getInstance().getLongitude());
                    if (!bPointInChina)
                    {
                        displayPVersion = true;
                    }
                 }
           }
           else {
               displayPVersion = true;
           }
        }
        if ((RegisterManage.GetInstance().getChinaLimitCode()&0x01) <=0 &&
              (RegisterManage.GetInstance().getValidFunction()&0x01) <=0 &&
              !displayPVersion)
        {
           setVisibility(R.id.labelDateTime, View.GONE);
           setVisibility(R.id.textViewDateTime, View.GONE);
           setVisibility(R.id.linearLayoutAuthorization, View.GONE);
        }
        else {
```

```
            setVisibility(R.id.labelDateTime, View.VISIBLE);
            setVisibility(R.id.textViewDateTime, View.VISIBLE);
        }
        setVisibility(R.id.linearLayout_PVersion, displayPVersion?View.VISIBLE:View.GONE);
        setTextViewText(R.id.textViewDeviceId, RegisterManage.GetInstance().getDeviceID());
        if (displayPVersion)
        {
            int expireDate = Math.min(RegisterManage.GetInstance().getExpireDate1(),
RegisterManage.GetInstance().getExpireDate2());
            int nYear = expireDate/10000;
            int nMonth = (expireDate%10000)/100;
            int nDay = expireDate%100;
            if(nYear<2000||nMonth==0||nDay==0)
                setTextViewText(R.id.textViewPDateTime, getString(R.string.SYS_FS_NOT_ENABLED));
            else {
                String strDateTime = nYear + "-" + nMonth + "-" + nDay;
                setTextViewText(R.id.textViewPDateTime, strDateTime);
            }
            if (expireDate - RegisterManage.GetInstance().getRealTimeDate() < 100 &&
                    RegisterManage.GetInstance().getExpireDate1() - expireDate > 0 &&
                    (overDateFunc&0x01) <= 0)
                setVisibility(R.id.button_Apply, View.VISIBLE);
            else
                setVisibility(R.id.button_Apply, View.GONE);
        }
        int expireDate = RegisterManage.GetInstance().getExpireDate1();
        if (!displayPVersion && ((overDateFunc&0x02) > 0 ||
(RegisterManage.GetInstance().getValidFunction()&0x02) > 0) &&
            expireDate != RegisterManage.GetInstance().getExpireDate2())
        {
            if (AppFinalUtil.isLanguageZhCh())
            {
                if (CustomGnssData.getInstance().isValidate() &&
                        CustomGnssData.getInstance().getSolutionType() !=
eGnssSolutionType.FIX_TYPE_INVALID)
                {
                    boolean bPointInChina =
RegisterManage.GetInstance().pointInChina(CustomGnssData.getInstance().getLatitude(),
CustomGnssData.getInstance().getLongitude());
                    if (!bPointInChina)
                    {
                        expireDate = Math.min(expireDate,
RegisterManage.GetInstance().getExpireDate2());
                    }
                }
            }
            else {
                expireDate = Math.min(expireDate, RegisterManage.GetInstance().getExpireDate2());
            }
        }
```

```
        int nYear = expireDate/10000;
        int nMonth = (expireDate%10000)/100;
        int nDay = expireDate%100;
        if(nYear<2000||nMonth==0||nDay==0) {
            setTextViewText(R.id.textViewDateTime, getString(R.string.SYS_FS_NOT_ENABLED));
            setVisibility(R.id.linearLayoutAuthorization, View.GONE);
        }
        else {
            String strDateTime = nYear + "-" + nMonth + "-" + nDay;
            String strRegisterKeyId = ConfigSystem.getInstance().getSoftwareRegisterKeyId();
            if (strRegisterKeyId.length()>8)
            {
                if (AppFinalUtil.getAppID() == eAppIdentifierType.APP_ID_SURVEY_GINTEC ||
                        AppFinalUtil.getAppID().isSouthVersion()) {
                    setVisibility(R.id.linearLayoutAuthorization, View.VISIBLE);
                    String strAuthorization = StringFunction.format("%s***%s",
strRegisterKeyId.substring(0, 2), strRegisterKeyId.substring(strRegisterKeyId.length() - 6));
                    setTextViewText(R.id.textViewAuthorization, strAuthorization);
                }
                else {
                    setVisibility(R.id.linearLayoutAuthorization, View.GONE);
                    strDateTime += StringFunction.format("(%s***%s)", strRegisterKeyId.substring(0,
2), strRegisterKeyId.substring(strRegisterKeyId.length()-6));
                }
            }
            else {
                setVisibility(R.id.linearLayoutAuthorization, View.GONE);
            }
            setTextViewText(R.id.textViewDateTime, strDateTime);
        }
    }
    @Override
    public void onConfigurationChanged (Configuration newConfig) {
        super.onConfigurationChanged(newConfig);
        setContentView(R.layout.activity_program_about);
        initialUI();
    }
    @Override
    public void onClick(View arg0)
    {
        switch (arg0.getId()) {
            case R.id.button_New_Version:
                RegisterAutoManageClient.getInstance().checkVersionUpdate();
                showWaitingDialog(true);
                break;
            case R.id.button_Apply:
            {
                startActivityForResult(new Intent(this, ApplyForRegisterCodeActivity.class),
R.id.button_Apply);
            }
```

```
            break;
        case R.id.button_Activation:
            startActivityForResult(new Intent(this, SoftwareActivateActivity.class),
R.id.button_Activation);
            break;
        case R.id.button_Activation_Trans_out:
            if (RegisterManage.GetInstance().getValidFunction() > 0)
            {
                new CustomDialog(this, getString(R.string.string_prompt),
                      getString(R.string.string_trans_out_register),
                      getString(R.string.button_ok),getString(R.string.button_cancel))
                      .setDialogListener(new CustomDialog.OnDialogListener() {
                         @Override
                         public void onPositiveClick(View customView, DialogInterface
dialogInterface, int which) {
                            showWaitingDialog(true);
                            transOutSocketClient.transOutActivation();
                         }
                         @Override
                         public void onNegativeClick(View customView, DialogInterface
dialogInterface, int which) {
                         }
                      }).showDialog();
            }
            else {
                makeTextString(getString(R.string.string_no_trans_out_register_info));
            }
            break;
        case R.id.button_Feedback:
            startActivity(new Intent(this, UserFeedbackActivity.class));
            break;
        case R.id.textViewDeviceId:
        {
            try {
                MultiFormatWriter writer = new MultiFormatWriter();
                String info = RegisterManage.GetInstance().getDeviceID();//输入的内容
                String contents=new String(info.getBytes("UTF-8"),"ISO-8859-1");
                BitMatrix matrix = writer.encode(contents, BarcodeFormat.CODE_128, 1000, 250);
                BarcodeEncoder encoder = new BarcodeEncoder();
                final Bitmap bitmap = encoder.createBitmap(matrix);
                LinearLayout linearLayout = new LinearLayout(this);
                linearLayout.setOrientation(LinearLayout.VERTICAL);
                TextView textView = new TextView(this);
                textView.setGravity(Gravity.CENTER_HORIZONTAL);
                textView.setText(info);
                textView.setTextSize(20);
                ImageView image = new ImageView(this);
                image.setAdjustViewBounds(true);
                image.setImageBitmap(bitmap);
                linearLayout.addView(image);
```

```
                linearLayout.addView(textView);
                new CustomDialog(this, linearLayout ,
                      getString(R.string.dialog_scan_code_128_title),
                      null,getString(R.string.button_ok))
                      .setDialogListener(new CustomDialog.OnDialogListener() {
                         @Override
                         public void onPositiveClick(View customView, DialogInterface
dialogInterface, int which) {
                         }
                         @Override
                         public void onNegativeClick(View customView, DialogInterface
dialogInterface, int which) {
                         }
                      }).showDialog();
             } catch (WriterException e) {
                e.printStackTrace();
             }catch (Exception e){
             }
          }
          break;
       }
    }
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
       super.onActivityResult(requestCode, resultCode, data);
       if (requestCode == R.id.button_Activation || requestCode == R.id.button_Apply)
       {
          updateActivationInfo();
       }
    }
    public void onEventMainThread(MainEvent.NetworkVersionConnectStatus result) {
       if (!result.isConnectSuccess())
       {
          makeTextString(getString(R.string.string_prompt_connect_server_overtime));
       }
       showWaitingDialog(false);
    }
    public void onEventMainThread(MainEvent.NetworkActivationUpdateStatus result) {
       updateActivationInfo();
    }
    public void onEventMainThread(MainEvent.NetworkTransOutActivationErrorStatus result) {
       if (result.getErrorCode() != eErrorCode.SUCCESS)
       {
          makeTextString(result.getErrorCode().getTitle());
       }
       showWaitingDialog(false);
       updateActivationInfo();
    }
}
```