

```
1 public class DeviceConnectActivity extends CommonEventBaseFragmentActivity implements
2 ViewPager.OnPageChangeListener
3 {
4     public static boolean gDeviceConnect = false;
5     private DeviceConnectFragment deviceConnectFragment = null;
6     private BluetoothNetworkFragment bluetoothNetworkFragment = null;
7     private CommonFragmentAdapter mFragmentAdapter;
8     @Override
9     protected void onCreate(Bundle savedInstanceState)
10    {
11        super.onCreate(savedInstanceState);
12        setContentView(R.layout.activity_device_connect);
13        gDeviceConnect = true;
14        initialUI();
15        if (!ConfigDeviceManage.GetInstance().getDeviceModelType().isSupportNfc()) {
16            CommonBaseActivity.mDeviceNFC.init(this);
17            CommonBaseActivity.mDeviceNFC.setNfcCallback(this);
18        }
19    }
20    private void initialUI()
21    {
22        mFragmentAdapter = new CommonFragmentAdapter(getSupportFragmentManager());
23        deviceConnectFragment = new DeviceConnectFragment();
24        mFragmentAdapter.addFragment(deviceConnectFragment);
25        if (AppFinalUtil.getAppID().isMiniApp())
26        {
27            bluetoothNetworkFragment = new BluetoothNetworkFragment();
28            mFragmentAdapter.addFragment(bluetoothNetworkFragment);
29        }
30        ViewPager viewPager = findViewById(R.id.viewPager_Fragment);
31        viewPager.setAdapter(mFragmentAdapter);
32        TabLayout tabLayout = findViewById(R.id.tabLayout_Fragment);
33        tabLayout.setupWithViewPager(viewPager);
34        if (mFragmentAdapter.getCount() <= 1)
35        {
36            tabLayout.setVisibility(View.GONE);
37        }
38        else {
39            if (DeviceManage.GetInstance().getConnectedStatus() ==
40 ConnectListener.CommanderStatus.SUCCESS)
41            {
42                viewPager.setCurrentItem(1);
43            }
44        }
45        CustomWaitingLayout waitingLayout = findViewById(R.id.waitingLayout);
46        waitingLayout.setOnClickListener(new View.OnClickListener() {
47            @Override
48            public void onClick(View v) {
49                DeviceManage.GetInstance().disconnect();
50                showWaitingDialog(false);
```

```
1     }
2     });
3     CustomCommandWaittingLayout          commandWaittingLayout          =
4     findViewById(R.id. commandWaittingLayout);
5     commandWaittingLayout.setOnCommandListener(new
6     CustomCommandWaittingLayout.OnCommandListener() {
7         @Override
8         public void onCommandStop() {
9             deviceConnectFragment.onButtonStop();
10        }
11        @Override
12        public void onCommandFinish(boolean finish) {
13            if (finish) {
14                finish();
15            }
16        }
17    });
18    }
19    protected void showWaitingDialog(boolean isShow)
20    {
21        setVisibility(R.id. waittingLayout, isShow ?View. VISIBLE : View. GONE);
22    }
23    @Override
24    public void onActivityResult(int requestCode, int resultCode, Intent data) {
25        super.onActivityResult(requestCode, resultCode, data);
26        ViewPager viewPager = findViewById(R.id. viewPager_Fragment);
27        mFragmentManager.getItem(viewPager.getCurrentItem()).onActivityResult((requestCode &
28    0xFFFF), resultCode, data);
29    }
30    @Override
31    public void onDeviceFound(String name, String address) {
32        DeviceModelType type = ConfigDeviceManage. GetInstance().getDeviceModelType();
33        if (deviceConnectFragment != null && type.isSupportNfc() &&
34    CommandSendManage. GetInstance().size() <= 0)
35        {
36            if (DeviceManage. GetInstance().isConnected())
37            {
38                if (DeviceManage. GetInstance().getConnectedStatus() !=
39    ConnectListener.CommanderStatus. FAIL &&
40                address.equals(ConfigDeviceManage. GetInstance().getBluetoothAddress()))
41                return;
42                deviceConnectFragment.onButtonStop();
43            }
44            ConfigDeviceManage. GetInstance().setBluetoothName(name);
45            ConfigDeviceManage. GetInstance().setBluetoothAddress(address);
46            deviceConnectFragment.updateDeviceLinker(DeviceLinkerType. BLUETOOTH);
47            deviceConnectFragment.onButtonConnect();
48        }
49    }
50
```

```
1 public void onEventMainThread(DeviceUpdateEvent.UpdateNetworkConnectStatus obj)
2 {
3     if (obj == null || bluetoothNetworkFragment == null)
4         return;
5     Toast toast = Toast.makeText(this, obj.getStatus(), Toast.LENGTH_SHORT);
6     toast.setGravity(Gravity.CENTER, 0, 0);
7     toast.show();
8     bluetoothNetworkFragment.updateConnectUI();
9 }
10 public void onEventMainThread(DeviceUpdateEvent.ReceiverNetworkData obj)
11 {
12     if (obj == null || bluetoothNetworkFragment == null)
13         return;
14     bluetoothNetworkFragment.receiverNetworkData(obj.getSize());
15 }
16 public void onEventMainThread(DeviceUpdateEvent.UpdateMountPointSuccessStatus obj) {
17     if (obj == null || bluetoothNetworkFragment == null)
18         return;
19     if (obj.IsGetMountPointSuccess()) {
20         SourecTableManage.getInstance().SaveSourceTable();
21         makeTextString(R.string.string_prompt_get_mount_point_list_succeeded,
22 Gravity.CENTER);
23     } else {
24         makeTextString(R.string.string_prompt_get_mount_point_failed, Gravity.CENTER);
25     }
26     showWaitingDialog(false);
27     bluetoothNetworkFragment.updateMountPointFinish();
28 }
29 public void onEventMainThread(DeviceUpdateEvent.DeviceConnectResultStatus eventStatus) {
30     showWaitingDialog(false);
31     if (eventStatus.getIsConnectSuccess())
32     {
33         DeviceManage.getInstance().setConnectState(true);
34         if (deviceConnectFragment != null)
35             deviceConnectFragment.updateButtonState();
36         CustomCommandWaittingLayout commandWaittingLayout =
37 findViewById(R.id.commandWaittingLayout);
38         commandWaittingLayout.startSendCommand(bluetoothNetworkFragment == null, true);
39     } else {
40         if (DeviceManage.getInstance().getCurLinkerType() == DeviceLinkerType.LOCAL) {
41             makeTextString(getString(R.string.toast_internal_gps_not_enabled));
42         } else {
43             makeTextString(getString(R.string.string_prompt_connection_failed));
44         }
45     }
46 }
47 @Override
48 public void finish()
49 {
50     if (deviceConnectFragment != null && deviceConnectFragment.cancelBatchSelectMode())
```

```
1         return;
2         CustomWaittingLayout waittingLayout = findViewById(R.id. waittingLayout);
3         if (waittingLayout.getVisibility() == View. VISIBLE)
4             return;
5         CustomCommandWaittingLayout commandWaittingLayout =
6         findViewById(R.id. commandWaittingLayout);
7         if (commandWaittingLayout.getVisibility() == View. VISIBLE)
8             return;
9         commandWaittingLayout.setOnCommandListener(null);
10        ConfigDeviceManage. GetInstance().saveConfig();
11        gDeviceConnect = false;
12        super.finish();
13    }
14    @Override
15    public void onPageScrolled(int position, float positionOffset, int positionOffsetPixels) {
16        if (bluetoothNetworkFragment != null) {
17            ViewPager viewPager = findViewById(R.id. viewPager_Fragment);
18            CommonV4Fragment commonV4Fragment =
19            mFragmentManager.getItem(viewPager.getCurrentItem());
20            CustomWaittingLayout waittingLayout = findViewById(R.id. waittingLayout);
21            if (commonV4Fragment == deviceConnectFragment)
22                {
23                    waittingLayout.setOnClickListener(new View.OnClickListener() {
24                        @Override
25                        public void onClick(View v) {
26                            DeviceManage. GetInstance().disconnect();
27                            showWaitingDialog(false);
28                        }
29                    });
30                }
31            else {
32                waittingLayout.setOnClickListener(null);
33            }
34        }
35    }
36    @Override
37    public void onPageSelected(int position) {
38    }
39    @Override
40    public void onPageScrollStateChanged(int state) {
41    }
42    }
43    public class BluetoothSearchActivity extends CommonGridBaseActivity
44        implements View.OnClickListener {
45        private DeviceModelType deviceModelType = DeviceModelType. TYPE_UNKNOWN;
46        private BroadcastReceiver mBluetoothReceiver = null;
47        private ArrayList<tagDeviceNode> deviceNodeArrayList = new ArrayList<>();
48        @Override
49        protected int getContentView() {
50            return R.layout. activity_bluetooth_search;
```

```
1     }
2     @Override
3     protected void initCustom() {
4         deviceModelType
5 DeviceModelType. swigToEnum(getIntent().getIntExtra("DeviceModelType",
6 DeviceModelType. TYPE_UNKNOWN.swigValue()));
7         try {
8             if (mGridItemListAdapter == null) {
9                 mGridItemListAdapter = new CustomGridBluetoothItemAdapter(this, this,
10 deviceNodeArrayList);
11             }
12         } catch (Exception e) {
13             e.printStackTrace();
14             return;
15         }
16         mGridView.setAdapter(mGridItemListAdapter);
17         if (mBluetoothReceiver == null) {
18             IntentFilter intentFilter = new IntentFilter();
19             intentFilter.addAction(BluetoothDevice. ACTION_FOUND);
20             intentFilter.addAction(BluetoothAdapter. ACTION_DISCOVERY_FINISHED);
21             intentFilter.addAction(BluetoothDevice. ACTION_BOND_STATE_CHANGED);
22             intentFilter.addAction(BluetoothAdapter. ACTION_SCAN_MODE_CHANGED);
23             intentFilter.addAction(BluetoothAdapter. ACTION_STATE_CHANGED);
24             mBluetoothReceiver = new BluetoothStateReceiver();
25             registerReceiver(mBluetoothReceiver, intentFilter);
26         }
27         updatePairedDeviceList();
28         onButtonSearchBluetooth();
29     }
30     @Override
31     public void onClickItem(int position) {
32         if (mGridItemListAdapter.isBatchSelectMode()) {
33             mGridItemListAdapter.onButtonCheckItem(position);
34         } else {
35             int selectPosition = position;
36             if (mGridItemListAdapter.getSelectedPosition() >= 0/* == position*/) {
37                 selectPosition = -1;
38             }
39             onClickApply(selectPosition);
40         }
41     }
42     private void updatePairedDeviceList() {
43         deviceNodeArrayList.clear();
44         for (int i = 0; deviceNodeArrayList.size() < 5 && i <
45 ConfigDeviceManage. GetInstance().getBluetoothDeviceCount(); i++) {
46             tagDeviceNode deviceNode = ConfigDeviceManage. GetInstance().getBluetoothDevice(i);
47             deviceNodeArrayList.add(deviceNode);
48         }
49         mGridItemListAdapter.setSelectedPosition(-1);
50         if (AppFinalUtil. getAppID().isXSurvey()) {
```

```
1         setVisibility(R.id. LinearLayout_BluetoothList, deviceNodeArrayList.size() > 0 ?
2 View. VISIBLE : View. GONE);
3     } else {
4         setVisibility(R.id. LinearLayout_BluetoothList, View. GONE);
5     }
6 }
7 @Override
8 public void finish() {
9     if (mBluetoothReceiver != null) {
10         unregisterReceiver(mBluetoothReceiver);
11         mBluetoothReceiver = null;
12     }
13     BluetoothAdapter adapter = BluetoothAdapter. getDefaultAdapter();
14     if (adapter != null && adapter.isDiscovering()) {
15         adapter.cancelDiscovery();
16     }
17     super.finish();
18 }
19 @Override
20 protected void onClickApply(int selectIndex) {
21     if (selectIndex < 0)
22         return;
23     tagDeviceNode deviceNode = (tagDeviceNode)mGridItemListAdapter.getItem(selectIndex);
24     finish(deviceNode.strName, deviceNode.strAddress);
25 }
26 @Override
27 protected void onButtonSelect() {
28 }
29 @Override
30 protected void onButtonAdd() {
31 }
32 @Override
33 protected void onButtonDelete(int index) {
34 }
35 @Override
36 protected void onButtonDelete(ArrayList<Integer> indexes) {
37     if (indexes.size() >= 5)
38     {
39         ConfigDeviceManage. GetInstance().removeAll();
40     }
41     else {
42         for (int i = indexes.size() - 1; i >= 0; i--) {
43             tagDeviceNode deviceNode = (tagDeviceNode)
44 mGridItemListAdapter.getItem(indexes.get(i));
45 ConfigDeviceManage. GetInstance().removeBluetoothDevice(deviceNode.strAddress);
46         }
47     }
48     updatePairedDeviceList();
49 }
50 @Override
```

```
1     protected void onButtonShare() {
2     }
3     private static void unpairDevice(BluetoothDevice device) {
4         try {
5             Method createBondMethod = device.getClass().getMethod("cancelBondProcess");
6             createBondMethod.invoke(device);
7         } catch (Exception e) {
8         }
9     }
10    @Override
11    protected void onButtonImport() {
12    }
13    @Override
14    protected void onButtonExport() {
15    }
16    @Override
17    protected void onButtonOK() {
18        onButtonSearchBluetooth();
19    }
20    private void onButtonSearchBluetooth() {
21        BluetoothAdapter adapter = BluetoothAdapter.getDefaultAdapter();
22        if (adapter == null) {
23            return;
24        }
25        if (adapter.isDiscovering()) {
26            adapter.cancelDiscovery();
27            setTextViewText(R.id.button_OK, getString(R.string.button_refresh));
28            return;
29        }
30        if (!checkGpsPermission()) {
31            makeTextString(R.string.toast_gps_permission_check);
32            return;
33        }
34        if (!checkOpen()) {
35            makeTextString(R.string.toast_turning_on_bluetooth);
36            return;
37        }
38        adapter.startDiscovery();
39        setTextViewText(R.id.button_OK, getString(R.string.string_prompt_searching));
40        CustomTextViewListLayout viewListLayoutAvailable
41    findViewById(R.id.viewListLayoutAvailable);
42        viewListLayoutAvailable.clear();
43    }
44    private boolean checkGpsPermission() {
45        if (android.os.Build.VERSION.SDK_INT >= 23) {
46            if (ContextCompat.checkSelfPermission(this,
47 Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
48                ActivityCompat.requestPermissions(this, new
49 String[] {Manifest.permission.ACCESS_FINE_LOCATION}, 101);
50                return false;
51            }
52        }
53    }
```

```
1         }
2         return true;
3     }
4     return true;
5 }
6 private boolean mBeWaitingWifiOpen = false;
7 private boolean checkOpen() {
8     if (mBeWaitingWifiOpen)
9         return false;
10    BluetoothAdapter mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
11    if (mBluetoothAdapter == null) {
12        return false;
13    }
14    if (!mBluetoothAdapter.isEnabled()) {
15        mBluetoothAdapter.enable();
16        mBeWaitingWifiOpen = true;
17        return false;
18    }
19    return true;
20 }
21 private void finish(String strName, String strAddress)
22 {
23     Intent intent = new Intent();
24     intent.putExtra("BluetoothName", strName);
25     intent.putExtra("BluetoothAddress", strAddress);
26     setResult(AppFinalUtil.RETURN_CODE_VALUE_BACK, intent);
27     finish();
28 }
29 @Override
30 public void onClickEdit() {
31 }
32 @Override
33 public void onClickShare() {
34 }
35 public class BluetoothStateReceiver extends BroadcastReceiver {
36     ArrayList<String> bluetoothDeviceList = new ArrayList<>();
37     @Override
38     public void onReceive(Context context, Intent intent) {
39         switch (intent.getAction()) {
40             case BluetoothDevice.ACTION_FOUND:
41                 BluetoothDevice device =
42 intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
43                 tagDeviceNode deviceNode = new tagDeviceNode();
44                 deviceNode.strName = device.getName();
45                 if (deviceNode.strName == null || deviceNode.strName.equals("")) {
46                     deviceNode.strName = "Unknow Device";
47                 }
48                 return;
49             if (deviceModelType == DeviceModelType.TYPE_MODEL_ZX_RTK)
50 {
```

```
1         if (deviceNode.strName.indexOf("Z") != 0)
2             return;
3     }
4     deviceNode.strAddress = device.getAddress();
5     LayoutInflater inflater = LayoutInflater.from(context);
6     View convertView =
7 inflater.inflate(R.layout.layout_gridview_bluetooth_item, null);
8     TextView textView = convertView.findViewById(R.id.textView_Name);
9     textView.setText(deviceNode.strName);
10    textView = convertView.findViewById(R.id.textView_Address);
11    textView.setText(deviceNode.strAddress);
12    convertView.setOnClickListener(new View.OnClickListener() {
13        @Override
14        public void onClick(View v) {
15            TextView textView = v.findViewById(R.id.textView_Name);
16            String strName = textView.getText().toString();
17            textView = v.findViewById(R.id.textView_Address);
18            String strAddress = textView.getText().toString();
19            finish(strName, strAddress);
20        }
21    });
22    CustomTextViewListLayout viewListLayoutAvailable =
23 findViewById(R.id.viewListLayoutAvailable);
24    View viewLine = convertView.findViewById(R.id.viewLine);
25    if (viewListLayoutAvailable.getCount() <= 0)
26    {
27        viewLine.setVisibility(View.GONE);
28        bluetoothDeviceList.clear();
29    }else {
30        viewLine.setVisibility(View.VISIBLE);
31    }
32    for (int i=0; i<bluetoothDeviceList.size(); i++)
33    {
34        if
35 (deviceNode.strAddress.equalsIgnoreCase(bluetoothDeviceList.get(i)))
36        {
37            return;
38        }
39    }
40    bluetoothDeviceList.add(deviceNode.strAddress);
41    viewListLayoutAvailable.add(convertView);
42    break;
43    case BluetoothAdapter.ACTION_DISCOVERY_FINISHED:
44        setTextViewText(R.id.button_OK, getString(R.string.button_refresh));
45        break;
46    case BluetoothAdapter.ACTION_STATE_CHANGED:
47        int blueState = intent.getIntExtra(BluetoothAdapter.EXTRA_STATE, 0);
48        switch(blueState) {
49            case BluetoothAdapter.STATE_TURNING_ON:
50                break;
```

```
1         case BluetoothAdapter.STATE_ON:
2             if (mBeWaitingWifiOpen) {
3                 mBeWaitingWifiOpen = false;
4                 onButtonSearchBluetooth();
5             }
6             break;
7         case BluetoothAdapter.STATE_TURNING_OFF:
8             break;
9         case BluetoothAdapter.STATE_OFF:
10            break;
11    }
12    break;
13    }
14    }
15    }
16    }
17    public class BluetoothNetworkFragment extends CommonV4Fragment implements OnClickListener
18    {
19        public tagDataLinkParameter dataLinkParameter = new tagDataLinkParameter();
20        @Nullable
21        @Override
22        public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, Bundle
23    savedInstanceState) {
24            if (mInfoView != null)
25                return mInfoView;
26            mInfoView = inflater.inflate(R.layout.layout_bluetooth_network, container, false);
27            dataLinkParameter.set(DeviceCommandParse.getInstance().mConfigWorkMode.dataLinkParameter);
28            DeviceManage.GetInstance().setPageInBlueToothSetting(true);
29            dataLinkParameter.bluetoothParam.set(ConfigNtripParameter.getInstance());
30            initializeUI();
31            updateValueToUI();
32            return mInfoView;
33        }
34        @Override
35        public String getTitle() {
36            return AppFinalUtil.getString(R.string.string_differential_data);
37        }
38        @Override
39        public void updateValueToUI()
40        {
41            if (mInfoView == null)
42                return;
43            CustomTextViewLayoutSelect layoutSelectDataLink =
44    mInfoView.findViewById(R.id.layoutSelect_DataLink);
45            layoutSelectDataLink.setSelectedId(dataLinkParameter.Type.swigValue());
46            updateConnectUI();
47        }
48        public void updateValueFormUI()
49        {
50            if (dataLinkParameter.Type == EDataLinkType.ExtendSource)
```



```
1         }
2         break;
3         case CORS_CONNECT_MODE_TCP: {
4             linearLayoutDataLink.add(2, StringFunction.format("%s:%s",
5 getString(R.string.string_server_ip), dataLinkParameter.bluetoothParam.strIP),
6 StringFunction.format("%s:%d",
7 dataLinkParameter.bluetoothParam.nPort), "", "");
8         }
9         break;
10        case CORS_CONNECT_MODE_ZHD://中海达
11        {
12            linearLayoutDataLink.add(2, StringFunction.format("%s:%s",
13 getString(R.string.string_server_ip), dataLinkParameter.bluetoothParam.strIP),
14 StringFunction.format("%s:%d",
15 dataLinkParameter.bluetoothParam.nPort), "", "");
16            linearLayoutDataLink.add(2, StringFunction.format("%s:%s",
17 getString(R.string.string_group_number), dataLinkParameter.bluetoothParam.strUser),
18 StringFunction.format("%s:%s",
19 dataLinkParameter.bluetoothParam.strPassword), "", "");
20        }
21        break;
22        case CORS_CONNECT_MODE_HUACE: {
23            linearLayoutDataLink.add(2, StringFunction.format("%s:%s",
24 getString(R.string.string_server_ip), dataLinkParameter.bluetoothParam.strIP),
25 StringFunction.format("%s:%d",
26 dataLinkParameter.bluetoothParam.nPort), "", "");
27            linearLayoutDataLink.add(1, StringFunction.format("%s:%s",
28 getString(R.string.string_huace_base_id), dataLinkParameter.bluetoothParam.strMountPoint),
29 "", "", "");
30        }
31        break;
32    }
33 }
34 break;
35 default:
36     break;
37 }
38 linearLayoutDataLink.setVisibility(bDisplay?View.VISIBLE: View.GONE);
39 setVisibility(R.id.linearLayout_MountPoint,
40 bNtripMountPoint?View.VISIBLE:View.GONE);
41 setVisibility(R.id.linearLayout_Cors, dataLinkParameter.Type ==
42 EDataLinkType.ExtendSource?View.VISIBLE:View.GONE);
43 }
44 });
45 layoutSelectDataLink.addItem(getString(R.string.title_bluetooth_dataLink), EDataLinkType.ExtendSource.swigValue());
46 CustomTextViewListLayout linearLayoutDataLink =
47 mInfoView.findViewById(R.id.linearLayout_DataLink);
48 linearLayoutDataLink.setOnClickListener(new OnClickListener() {
49     @Override
```

```

1      public void onClick(View v) {
2          Intent intent = new Intent(getContext(), EditRoverDataLinkParamActivity.class);
3          intent.putExtra("DataLinkType", dataLinkParameter.Type.swigValue());
4          switch (dataLinkParameter.Type)
5          {
6              case ExtendSource:
7                  {
8                      intent.putExtra("BluetoothParam",
9 dataLinkParameter.bluetoothParam.toString());
10                     }
11                     break;
12                 }
13                 startActivityForResult(intent, R.id.LinearLayout_DataLink&0xFFFF);
14             }
15         });
16     }
17     private void initializeMountPointList()
18     {
19         final CustomTextViewLayoutSelectEdit MountPoint =
20 mInfoView.findViewById(R.id.layoutSelectEdit_MountPoint);
21         MountPoint.setCustomClickListener(new OnClickListener() {
22             @Override
23             public void onClick(View v) {
24                 CustomInputActivity.InputListener inputListener = new
25 CustomInputActivity.InputListener() {
26                 @Override
27                 public void onSelect(String label, int index) {
28                     if (label.indexOf('(') > 0 &&
29 label.indexOf(')') > 0) {
30                         label = label.substring(0, label.indexOf('('));
31                     }
32                     MountPoint.setText(label);
33                 }
34             };
35             CustomInputActivity.SortChangeListener sortChangeListener = new
36 CustomInputActivity.SortChangeListener() {
37                 @Override
38                 public void onSortTypeChange(int index) {
39                     SourecTableManage.getInstance().setSortType(index);
40                 }
41                 @Override
42                 public ArrayList<String> getValueList() {
43                     return SourecTableManage.getInstance().getMountPointList();
44                 }
45                 @Override
46                 public String getValue(String inputOld) {
47                     return inputOld;
48                 }
49                 @Override
50                 public ArrayList<String> getSortTypeList() {

```

```
1         ArrayList<String> listSort = new ArrayList<>();
2         listSort.add(getString(R.string. label_sort_name_default));
3         listSort.add(getString(R.string. label_sort_name_pos));
4         listSort.add(getString(R.string. label_sort_name_neg));
5         if (SourecTableManage. getInstance(). isSurportSortByDist()) {
6             listSort.add(getString(R.string. label_sort_dist_pos));
7         }
8         return listSort;
9     }
10    @Override
11    public int getSortIndex() {
12        return SourecTableManage. getInstance(). getSortType();
13    }
14 };
15    new CustomInputActivity. Builder(getContext())
16        . setMode(CustomInputActivity. MODE_EDIT_SELECT)
17        . setList(SourecTableManage. getInstance(). getMountPointList())
18        . setValue(MountPoint. getText())
19        . setTitle(getString(R.string. string_ntrip_mount_point))
20        . setListener(inputListener)
21        . setSortChangeListener(sortChangeListener)
22        . start();
23    }
24 });
25 }
26 @Override
27 public void onClick(View v)
28 {
29     switch(v. getId())
30     {
31         case R. id. button_MountPoint:
32             {
33                 onButtonGetMountPoint();
34             }
35             break;
36         case R. id. button_Start:
37             {
38                 onButtonStart();
39             }
40             break;
41         case R. id. button_Stop:
42             {
43                 onButtonStop();
44             }
45             break;
46     }
47 }
48 @Override
49 public void onActivityResult(int requestCode, int resultCode, Intent data)
50 {
```

```

1      super.onActivityResult(requestCode, resultCode, data);
2      if(resultCode == AppFinalUtil.RETURN_CODE_VALUE_BACK)
3      {
4          if (requestCode == (R.id.linearlayout_DataLink&0xFFFF) && data != null)
5          {
6              switch (dataLinkParameter.Type){
7                  case ExtendSource:
8                      dataLinkParameter.bluetoothParam.parseValue(data.getStringExtra("BluetoothParam"));
9                      dataLinkParameter.bluetoothParam.strMountPoint =
10                     getViewString(R.id.layoutSelectEdit_MountPoint);
11                      break;
12              }
13              updateValueToUI();
14          }
15      }
16  }
17  private void onButtonGetMountPoint() {
18      if (mInfoView == null)
19          return;
20      mnTimeCount = 0;
21      mHandler.postDelayed(mRunnable, 1000);
22      tagConnectionParameter connectionParameter = new tagConnectionParameter();
23      connectionParameter.set(dataLinkParameter.bluetoothParam);
24      CorsClientManage.GetInstance().setNetworkParameter(connectionParameter.connectMode,
25 connectionParameter.strIP, connectionParameter.nPort, connectionParameter.strUser,
26 connectionParameter.strPassword);
27      CorsClientManage.GetInstance().onUpdateMountPoint();
28      CustomWaittingLayout waittingLayout =
29 AppFinalUtil.activityCurrent.findViewById(R.id.waittingLayout);
30      if (waittingLayout != null) {
31          waittingLayout.setLabel(getString(R.string.command_function_set_start_get_mount_point));
32          waittingLayout.setVisibility(View.VISIBLE);
33      }
34  }
35  public void onButtonStart() {
36      ConfigNtripParameter.GetInstance().setAutoConnect(true);
37      if(ConnectListener.CommanderStatus.SUCCESS !=
38 DeviceManage.GetInstance().getConnectedStatus())
39      {
40          Toast toast = Toast.makeText(getContext(),
41 R.string.string_prompt_communication_not_connected, Toast.LENGTH_SHORT);
42          toast.setGravity(Gravity.CENTER, 0, 0);
43          toast.show();
44          return;
45      }
46      if (CorsClientManage.GetInstance().isConnectCors())
47      {
48          return;
49      }
50      dataLinkParameter.bluetoothParam.strMountPoint =

```

```
1  getTextViewString(R.id. layoutSelectEdit_MountPoint);
2      ConfigNtripParameter. getInstance(). set (dataLinkParameter. bluetoothParam);
3      ConfigNtripParameter. getInstance(). saveConfig();
4  CorsClientManage. GetInstance(). setNetworkParameter (dataLinkParameter. bluetoothParam. getMode
5  (eDeviceWorkingMode. Rover),
6      dataLinkParameter. bluetoothParam. strIP, dataLinkParameter. bluetoothParam. nPort,
7      dataLinkParameter. bluetoothParam. strUser,
8  dataLinkParameter. bluetoothParam. strPassword,
9      dataLinkParameter. bluetoothParam. strMountPoint);
10  CorsClientManage. GetInstance(). startConnectCors();
11  updateConnectUI();
12  }
13  public void onButtonStop()
14  {
15      ConfigNtripParameter. getInstance(). setAutoConnect (false);
16      if (!CorsClientManage. GetInstance(). isConnectCors())
17      {
18          return;
19      }
20      CorsClientManage. GetInstance(). stopConnectCors();
21      updateConnectUI();
22  }
23  public void updateConnectUI()
24  {
25      if (mInfoView == null)
26          return;
27      View viewCors = mInfoView. findViewById(R.id. linearLayout_Cors);
28      boolean bConnect = CorsClientManage. GetInstance(). isConnectCors();
29      if (viewCors. getVisibility() != View. VISIBLE)
30      {
31          bConnect = false;
32      }
33      setVisibility(R.id. button_Start, bConnect?View. GONE:View. VISIBLE);
34      setVisibility(R.id. button_Stop, bConnect?View. VISIBLE:View. GONE);
35      setEnabled(R.id. button_MountPoint, !bConnect);
36      setEnabled(R.id. layoutSelectEdit_MountPoint, !bConnect);
37      setEnabled(R.id. layoutSelect_DataLink, false/*!bConnect*/);
38      setEnabled(R.id. linearLayout_DataLink, !bConnect);
39  }
40  int mnCountReceiver = 0;
41  int nLength = 0;
42  long currentTime = 0;
43  public void receiverNetworkData(int length)
44  {
45      if (currentTime/1000 == System. currentTimeMillis()/1000)
46      {
47          nLength += length;
48      }
49      else {
50          currentTime = System. currentTimeMillis();
```

```
1         nLength = length;
2     }
3     Message message = new Message();
4     Bundle bundle = new Bundle();
5     bundle.putInt("length", nLength);
6     message.what = 0;
7     message.setData(bundle);
8     mHandler.sendMessage(message);
9 }
10 public void updateMountPointFinish()
11 {
12     mnTimeCount = 61;
13 }
14 private Handler mHandler = new Handler()
15 {
16     public void handleMessage(Message msg)
17     {
18         switch (msg.what)
19         {
20             case 0: {
21                 View viewCors = mInfoView.findViewById(R.id.linearLayout_Cors);
22                 if (viewCors.getVisibility() != View.VISIBLE)
23                     return;
24                 int length = msg.getData().getInt("length");
25                 CustomTimerView timerView_Rev = mInfoView.findViewById(R.id.timerView_Rev);
26                 mnCountReceiver += length;
27                 mnCountReceiver = (mnCountReceiver%8000);
28                 timerView_Rev.setPosValue(mnCountReceiver);
29                 timerView_Rev.setPromptTextString(String.format("%dB", length));
30             }
31             break;
32         }
33     }
34 };
35 int mnTimeCount = 0;
36 Runnable mRunnable = new Runnable()
37 {
38     @Override
39     public void run()
40     {
41         mnTimeCount++;
42         if (mnTimeCount == 60) {
43             DeviceUpdateUINotify.eventUpdateMountPointSuccess(false);
44         }
45         if (mnTimeCount < 60) {
46             mHandler.postDelayed(this, 1000);
47         }
48     }
49 };
50 @Override
```

```
1 public void onDestroy() {
2     DeviceManage. GetInstance(). setPageInBlueToothSetting(false);
3     mHandler.removeCallbacks(mRunnable);
4     super.onDestroy();
5 }
6 }
7 public class ProjectCreateActivity extends CommonEventBaseFragmentActivity
8     implements OnClickListener, ViewPager.OnPageChangeListener
9 {
10     private ProjectBasicInfoFragment basicInfoFragment = null;
11     private ProjectCoordinateSystemFragment coordinateSystemFragment = null;
12     private CommonFragmentAdapter mFragmentAdapter;
13     @Override
14     protected void onCreate(Bundle savedInstanceState)
15     {
16         super.onCreate(savedInstanceState);
17         setContentView(R.layout. activity_project_create);
18         initialUI();
19     }
20     private void initialUI()
21     {
22         initClickListener(R.id. button_OK, this);
23         initClickListener(R.id. button_Pre, this);
24         initClickListener(R.id. button_Next, this);
25         initClickListener(R.id. button_Cancel, this);
26         mFragmentAdapter = new CommonFragmentAdapter(getSupportFragmentManager());
27         basicInfoFragment = new ProjectBasicInfoFragment();
28         coordinateSystemFragment = new ProjectCoordinateSystemFragment();
29         mFragmentAdapter.addFragment(basicInfoFragment);
30         mFragmentAdapter.addFragment(coordinateSystemFragment);
31         ViewPager viewPager = findViewById(R.id. viewPager_Fragment);
32         viewPager.setAdapter(mFragmentAdapter);
33         viewPager.addOnPageChangeListener(this);
34         TabLayout tabLayout = findViewById(R.id. tabLayout_Fragment);
35         tabLayout.setupWithViewPager(viewPager);
36     }
37     @Override
38     public void onPageScrolled(int i, float v, int i1) {
39         setVisibility(R.id. button_Cancel, i==0?View.VISIBLE:View.GONE);
40         setVisibility(R.id. button_Next, i==0?View.VISIBLE:View.GONE);
41         setVisibility(R.id. button_Pre, i==1?View.VISIBLE:View.GONE);
42         setVisibility(R.id. button_OK, i==1?View.VISIBLE:View.GONE);
43         if (basicInfoFragment == null || coordinateSystemFragment == null)
44             return;
45         if (i==1)
46         {
47             if (basicInfoFragment.isUseLastCoordinateSystemParam())
48             {
49                 coordinateSystemFragment.setCoordSystemType(ProjectManage. GetInstance(). getUseCoordSystemType());
50             }
51         }
52     }
53 }
```

```
1         if (coordinateSystemFragment.getCoordSystemType() ==
2 eCoordinateSystemType. SYSTEM_TYPE_LOCAL)
3         {
4 coordinateSystemFragment.setCoordinateSystemParameter(ProjectCoordSystem.GetInstance().getC
5 oordSystemParameter());
6         }
7         else if (coordinateSystemFragment.getCoordSystemType() ==
8 eCoordinateSystemType. SYSTEM_TYPE_RTCM)
9         {
10 coordinateSystemFragment.setRtcmCoordinateSystemParameter(ProjectCoordSystem.GetInstance().
11 getRtcmCoordSystemParameter());
12         }
13     }
14     else {
15 coordinateSystemFragment.setCoordSystemType(eCoordinateSystemType. SYSTEM_TYPE_LOCAL);
16         tagCoordinateSystemParameter systemParameter = new tagCoordinateSystemParameter();
17         if(AppFinalUtil.isLanguageZhCh())
18         {
19             systemParameter.setName("CGCS2000");
20             tagEllipsoidParameter ellipsoidParameter = new tagEllipsoidParameter();
21             ellipsoidParameter.setName("CGCS2000");
22             ellipsoidParameter.setDf(298.257222101);
23             ellipsoidParameter.setDa(6378137);
24             systemParameter.setEllipsoidParam(ellipsoidParameter);
25         }
26         else {
27             systemParameter.setName("Default");
28         }
29         coordinateSystemFragment.setCoordinateSystemParameter(systemParameter);
30     }
31 }
32 }
33 @Override
34 public void onPageSelected(int i) {
35 }
36 @Override
37 public void onPageScrollStateChanged(int i) {
38 }
39 @Override
40 public void onClick(View v) {
41     switch (v.getId()) {
42         case R.id.button_OK: {
43             if (!basicInfoFragment.testProjectName())
44             {
45                 ViewPager viewPager = findViewById(R.id.viewPager_Fragment);
46                 viewPager.setCurrentItem(0);
47                 basicInfoFragment.setTextViewFocus(R.id.editText_ProjectName);
48                 return;
49             }
50             if (!createProject())
```

```
1      {
2          return;
3      }
4      setResult(AppFinalUtil.RETURN_CODE_DEFAULT);
5      finish();
6  }
7  break;
8  case R.id.button_Pre:
9  {
10     ViewPager viewPager = findViewById(R.id.viewPager_Fragment);
11     viewPager.setCurrentItem(0);
12 }
13 break;
14 case R.id.button_Next:
15 {
16     ViewPager viewPager = findViewById(R.id.viewPager_Fragment);
17     viewPager.setCurrentItem(1);
18 }
19 break;
20 case R.id.button_Cancel:
21 {
22     finish();
23 }
24 break;
25 default:
26     break;
27 }
28 }
29 @Override
30 public void onActivityResult(int requestCode, int resultCode, Intent data) {
31     ViewPager viewPager = findViewById(R.id.viewPager_Fragment);
32     mFragmentManager.getItem(viewPager.getCurrentItem()).onActivityResult(requestCode&0xFFFF,
33     resultCode, data);
34 }
35 private boolean createProject()
36 {
37     String strProjectPath = basicInfoFragment.getProjectPath();
38     String strProjectName = basicInfoFragment.getProjectName();
39     String strOperator = basicInfoFragment.getOperator();
40     String strRemark = basicInfoFragment.getRemark();
41     String strCreateTime = basicInfoFragment.getCreateTime();
42     eCoordinateSystemType systemType = coordinateSystemFragment.getCoordSystemType();
43     tagCoordinateSystemParameter coordinateSystemParameter =
44     coordinateSystemFragment.getCoordinateSystemParameter();
45     BaseChangeCorrectParameter correctParameter = new BaseChangeCorrectParameter();
46     tagGnssRefStationItem refStationItem = null;
47     if (basicInfoFragment.isRetainBaseCorrectParam())
48     {
49         correctParameter.set(ConfigBaseCorrect.getInstance());
50         refStationItem =
```

```
1 ProjectDbFileManage. getInstance().getRefStationData(correctParameter.strBaseKeyId);
2     }
3     if (!ProjectManage. GetInstance().createProject(strProjectPath, strProjectName,
4         strOperator, strRemark, strCreateTime, systemType, coordinateSystemParameter))
5     {
6         return false;
7     }
8     if (systemType == eCoordinateSystemType. SYSTEM_TYPE_RTCM)
9     {
10    ProjectCoordSystem. GetInstance().setRtcmCoordSystemParameter(coordinateSystemFragment.getRt
11    cmCoordinateSystemParameter());
12    }
13    if (refStationItem != null)
14    {
15        ProjectDbFileManage. getInstance().insertRefStationInfo(refStationItem);
16    }
17    if (correctParameter.isValid())
18    {
19        ProjectDbFileManage. getInstance().insertBaseCorrectParameter(correctParameter);
20    }
21    if (!correctParameter.isEquals(ConfigBaseCorrect. getInstance())) {
22        ConfigBaseCorrect. getInstance().set(correctParameter);
23        ConfigBaseCorrect. getInstance().saveConfig();
24    }
25    return true;
26 }
27 public void onEventMainThread(MainEvent.UpdateRtcmParam obj) {
28     tagRtcmCoordinateSystemParameter          coordinateSystemParameter          =
29    coordinateSystemFragment.getRtcmCoordinateSystemParameter();
30     tagRtcmCoordinateSystemParameter          rtcmSystemParameter          =
31    ProjectCoordSystem. GetInstance().getRtcm31Convert().getRtcmCoordinateSystemParameter();
32     if (coordinateSystemParameter.getUseSrcEllipsoidRtcm())
33     {
34        coordinateSystemParameter.setSrcEllipsoidVaild(rtcmSystemParameter.getSrcEllipsoidVaild());
35        coordinateSystemParameter.setSrcEllipsoidParam(rtcmSystemParameter.getSrcEllipsoidParam());
36    }
37     if (coordinateSystemParameter.getUseTargetEllipsoidRtcm())
38     {
39        coordinateSystemParameter.setTargetEllipsoidVaild(rtcmSystemParameter.getTargetEllipsoidVai
40    ld());
41        coordinateSystemParameter.setTargetEllipsoidParam(rtcmSystemParameter.getTargetEllipsoidPar
42    am());
43    }
44     if (coordinateSystemParameter.getUseProjectRtcm())
45     {
46        coordinateSystemParameter.setProjectVaild(rtcmSystemParameter.getProjectVaild());
47        coordinateSystemParameter.setProjectParam(rtcmSystemParameter.getProjectParam());
48    }
49     if (coordinateSystemParameter.getUseServerRtcm())
50     {
```

```
1         coordinateSystemParameter.setSerVenVaild(rtcMSystemParameter.getSerVenVaild());
2         coordinateSystemParameter.setDatumParam(rtcMSystemParameter.getDatumParam());
3     }
4     if (coordinateSystemParameter.getUseGridRtcM())
5     {
6         coordinateSystemParameter.setGridVaild(rtcMSystemParameter.getGridVaild());
7         coordinateSystemParameter.setGridParam(rtcMSystemParameter.getGridParam());
8     }
9     coordinateSystemFragment.setRtcMCoordinateSystemParameter(coordinateSystemParameter);
10    coordinateSystemFragment.updateValueToUI();
11    }
12 }
13 public class ProjectManage extends ProjectConfig
14 {
15     private static ProjectManage mProjectManage = null;
16     public static ProjectManage GetInstance()
17     {
18         if (null == mProjectManage)
19         {
20             mProjectManage = new ProjectManage();
21             mProjectManage.initializeXSurvey();
22         }
23         return mProjectManage;
24     }
25     private void initializeXSurvey()
26     {
27         mstrProgramPath = getDefaultProgramPath();
28         loadProjectConfig();
29         IoFileManager.createDirectory(mstrProgramPath);
30         mstrLastProjectPath = getProgramProjectPath();
31         IoFileManager.createDirectory(mstrLastProjectPath);
32         IoFileManager.createDirectory(getProgramConfigPath());
33         IoFileManager.createDirectory(getProgramDebugPath());
34         IoFileManager.createDirectory(getProgramImportPath());
35         IoFileManager.createDirectory(getProgramExportPath());
36         IoFileManager.createDirectory(getProgramMapPath());
37         IoFileManager.createDirectory(getProgramDataPath());
38         IoFileManager.createDirectory(getProgramGeoidPath());
39         IoFileManager.createDirectory(getProgramInstallationPath());
40     }
41     public boolean loadLastProject()
42     {
43         if (mProjectItemList.size() > 0)
44             return openProject(mProjectItemList.get(0).strProjectFilePath);
45         return false;
46     }
47     public void removeProjectItem(String strProjectFilePath)
48     {
49         for (int i = mProjectItemList.size()-1; i >= 0; i--)
50         {
```

```
1         if (mProjectItemList.get(i).strProjectFilePath.equalsIgnoreCase(strProjectFilePath))
2         {
3             mProjectItemList.remove(i);
4             saveLastProjectName();
5         }
6     }
7 }
8 public String getProjectPath()
9 {
10     return mstrLastProjectPath;
11 }
12 public boolean createProject(String strProjectPath, String strProjectName, String
13 strOperator, String strRemark, String strCreateTime,
14 eCoordinateSystemType systemType, tagCoordinateSystemParameter
15 systemParameter)
16 {
17     File temFile = new File(strProjectPath + "/" + strProjectName);
18     if (temFile.exists())
19     {
20         return false;
21     }
22     IoFileManage.createDirectory(strProjectPath + "/" + strProjectName);
23     closeProject();
24     setOperator(strOperator);
25     setRemark(strRemark);
26     setCreateTime(strCreateTime);
27     String strProjectPathName = strProjectPath + "/" + strProjectName + "/config.job";
28     if (!saveConfig(strProjectPathName))
29     {
30         return false;
31     }
32     misOpenProject = true;
33     mstrLastProjectPath = strProjectPath;
34     mstrProjectName = strProjectName;
35     IoFileManage.createDirectory(getProjectDataDirectory());
36     IoFileManage.createDirectory(getProjectConfigDirectory());
37     String strDataFilePath = getProjectDirectory() + "/" + "surveydata.db";
38     ProjectDbFileManage.getInstance().createSQLiteDB(strDataFilePath);
39     setUseCoordSystemType(systemType);
40     ProjectCoordSystem.GetInstance().setConfigPath(getProjectDirectory() + "/" +
41 "coordparam.sys");
42     if (systemType == eCoordinateSystemType.SYSTEM_TYPE_LOCAL) {
43         ProjectCoordSystem.GetInstance().setCoordSystemParameter(systemParameter);
44         ProjectCoordSystem.GetInstance().Save();
45     }
46     loadConfig();
47     ConfigConvertParam.getInstance().initConfig();
48     ConfigStakeoutPoint.getInstance().initConfig();
49     tagProjectItem projectItem = new tagProjectItem();
50     projectItem.strProjectName = strProjectName;
```

```
1     projectItem.strProjectFilePath = strProjectPathName;
2     projectItem.strLastTime      =      StringFunction.formatDate("yyyy-MM-dd      HH:mm:ss",
3 Calendar.getInstance().getTime());
4     projectItem.strCreateTime = getCreateTime();
5     projectItem.strOperator = getOperator();
6     if (mProjectItemList.size() > 0)
7         mProjectItemList.add(0, projectItem);
8     else
9         mProjectItemList.add(projectItem);
10    saveLastProjectName();
11    return true;
12    }
13    public boolean renameProject(String strProjectName)
14    {
15        String strTemProjectName = getProjectName();
16        File temFile = new File(getProjectPath() + "/" + strProjectName);
17        if (temFile.exists())
18            {
19                return false;
20            }
21        ProjectDbFileManage.getInstance().closeSQLiteDB();
22        File oldFile = new File(getProjectPath() + "/" + strTemProjectName);
23        oldFile.renameTo(temFile);
24        mstrProjectName = strProjectName;
25        oldFile      =      new      File(StringFunction.format("%s/%s.bck",      getProgramDataPath(),
26 strTemProjectName));
27        if (oldFile.exists()){
28            temFile      =      new      File(StringFunction.format("%s/%s.bck",      getProgramDataPath(),
29 strProjectName));
30            oldFile.renameTo(temFile);
31        }
32        String strDataFilePath = getProjectDirectory() + "/" + "surveydata.db";
33        ProjectDbFileManage.getInstance().openSQLiteDB(strDataFilePath);
34        String strProjectPathName = getProjectPath() + "/" + strTemProjectName + "/config.job";
35        for (int i = mProjectItemList.size()-1; i >= 0; i--)
36            {
37                if (mProjectItemList.get(i).strProjectFilePath.equalsIgnoreCase(strProjectPathName))
38                    {
39                        tagProjectItem projectItem = mProjectItemList.get(i);
40                        strProjectPathName = getProjectPath() + "/" + strProjectName + "/config.job";
41                        projectItem.strProjectName = mstrProjectName;
42                        projectItem.strProjectFilePath = strProjectPathName;
43                        projectItem.strLastTime      =      StringFunction.formatDate("yyyy-MM-dd      HH:mm:ss",
44 Calendar.getInstance().getTime());
45                        saveLastProjectName();
46                    }
47            }
48        return true;
49    }
50    public boolean openProject(String strProjectPathName)
```

```
1      {
2          if (!loadConfig(strProjectPathName))
3              return false;
4          strProjectPathName = strProjectPathName.replace("//", "/");
5          String strTem = strProjectPathName.substring(0, strProjectPathName.lastIndexOf('/'));
6          mstrLastProjectPath = strTem.substring(0, strTem.lastIndexOf('/'));
7          mstrProjectName = strTem.substring(strTem.lastIndexOf('/') + 1);
8          IoFileManage.createDirectory(getProjectDataDirectory());
9          String strDataFilePath = getProjectDirectory() + "/" + "surveydata.db";
10         ProjectDbFileManage.getInstance().openSQLiteDB(strDataFilePath);
11         setUseEncrypt(isCoordinateSystemParEncrypt());
12         loadConfig();
13         ProjectCoordSystem.GetInstance().setUseCoordSystemType(getUseCoordSystemType());
14         IoFileManage.createDirectory(getProjectConfigDirectory());
15         removeProjectItem(strProjectPathName);
16         tagProjectItem projectItem = new tagProjectItem();
17         projectItem.strProjectName = mstrProjectName;
18         projectItem.strProjectFilePath = strProjectPathName;
19         projectItem.strLastTime = StringFunction.formatDate("yyyy-MM-dd HH:mm:ss",
20 Calendar.getInstance().getTime());
21         projectItem.strCreateTime = getCreateTime();
22         projectItem.strOperator = getOperator();
23         if (mProjectItemList.size() > 0)
24             mProjectItemList.add(0, projectItem);
25         else
26             mProjectItemList.add(projectItem);
27         saveLastProjectName();
28         misOpenProject = true;
29         return true;
30     }
31     public void setUseEncrypt(boolean bEncrypt)
32     {
33         setCoordinateSystemParEncrypt(bEncrypt);
34         Save();
35         ProjectCoordSystem.GetInstance().loadConfig(getProjectDirectory() + "/coordparam.sys");
36     }
37     public void setUseCoordSystemType(eCoordinateSystemType nType) {
38         super.setUseCoordSystemType(nType);
39         if (isOpenProject()) {
40             Save();
41         }
42         ProjectCoordSystem.GetInstance().setUseCoordSystemType(nType);
43     }
44     public boolean InputPassword(String strPassword)
45     {
46         return ProjectCoordSystem.GetInstance().inputPassword(strPassword);
47     }
48     @Override
49     public void setOperator(String strOperator) {
50         if (isOpenProject())
```

```
1      {
2          if (mProjectItemList.size() > 0)
3              mProjectItemList.get(0).strOperator = strOperator;
4              saveLastProjectName();
5          }
6          super.setOperator(strOperator);
7      }
8      public void Save()
9      {
10         String strProjectPathName = mstrLastProjectPath + "/" + mstrProjectName + "/config.job";
11         saveConfig(strProjectPathName);
12     }
13     private void loadConfig()
14     {
15         ConfigConvertParam.getInstance().loadConfig();
16         ConfigRecord.getInstance().loadConfig();
17         ConfigMapConvert.getInstance().loadConfig();
18         ConfigCadEdit.getInstance().loadConfig();
19         ConfigStakeoutPoint.getInstance().loadConfig();
20         ConfigDisplayInfoManage.getInstance().loadConfig();
21     }
22     public void closeProject()
23     {
24         String strDataFilePath = getProjectDirectory() + "/" + "surveydata.db";
25         ProjectDbFileManage.getInstance().closeSQLiteDB();
26         MediaScannerConnection.scanFile(AppFinalUtil.activityCurrent,
27             new String[] { strDataFilePath }, null, null);
28         misOpenProject = false;
29         mstrProjectName = "";
30         clear();
31     }
32     public String getProjectName() {
33         return mstrProjectName;
34     }
35     public boolean isOpenProject()
36     {
37         return misOpenProject;
38     }
39     public ArrayList<tagProjectItem> getProjectItemList()
40     {
41         return mProjectItemList;
42     }
43     public String getProjectDirectory()
44     {
45         return getProjectPath() + "/" + mstrProjectName;
46     }
47     public String getProjectDataDirectory()
48     {
49         return getProjectDirectory() + "/Data";
50     }
51 }
```

```
1 public String getProjectConfigDirectory()
2 {
3     return getProjectDirectory() + "/Config";
4 }
5 public String getProjectGisImagesDirectory() {
6     return getProjectDirectory() + "/Images";
7 }
8 public String getProjectGisFileDirectory() {
9     return getProjectDirectory() + "/Files";
10 }
11 public String getDefaultProgramPath()
12 {
13     String strPathProgram = Environment.getExternalStorageDirectory().getAbsolutePath() +
14 AppFinalUtil.getAppID().getAppDirectory();
15     return strPathProgram;
16 }
17 public String getProgramProjectPath()
18 {
19     String strProjectPath = getProjectPath();
20     if (strProjectPath == null || strProjectPath.length() <= 0)
21     {
22         strProjectPath = mstrProgramPath + "/Project";
23     }
24     return strProjectPath;
25 }
26 public String getProgramConfigPath()
27 {
28     return mstrProgramPath + "/Config";
29 }
30 public String getProgramDebugPath()
31 {
32     return mstrProgramPath + "/Debug";
33 }
34 public String getProgramDataPath()
35 {
36     return mstrProgramPath + "/Data";
37 }
38 public String getProgramImportPath()
39 {
40     return mstrProgramPath + "/Import";
41 }
42 public String getProgramExportPath()
43 {
44     return mstrProgramPath + "/Export";
45 }
46 public String getProgramMapPath()
47 {
48     return mstrProgramPath + "/Map";
49 }
50 public String getProgramGeoidPath() {
```

```
1     return mstrProgramPath + "/Geoid";
2 }
3 public String getProgramInstallationPath() {
4     return mstrProgramPath + "/Installation";
5 }
6 public String getProgramCachePath() {
7     return mstrProgramPath + "/.Cache";
8 }
9 private void loadProjectConfig()
10 {
11     mProjectItemList.clear();
12     String programConfigFile = getProgramConfigPath() + "/projectInfo.ini";
13     CommonFile temFile = new CommonFile(programConfigFile);
14     CBufferData temBufferData = temFile.openReadAll(CommonFile.FORMAT_UTF8);
15     if (temBufferData == null)
16         return;
17     CParseString temParseString = new CParseString();
18     String strLine;
19     while ((strLine = temBufferData.ReadLine()) != null) {
20         if (strLine.isEmpty())
21             continue;
22         if(strLine.contains(", "))
23             strLine = strLine.replaceAll(", ", ",");
24         int nCount = temParseString.split(strLine, ",");
25         if (nCount < 4) {
26             continue;
27         }
28         tagProjectItem item = new tagProjectItem();
29         item.strProjectName = temParseString.getValueString(0);
30         item.strProjectFilePath = temParseString.getValueString(1);
31         item.strLastTime = temParseString.getValueString(2);
32         item.strCreateTime = temParseString.getValueString(3);
33         item.strOperator = temParseString.getValueString(4);
34         mProjectItemList.add(item);
35     }
36 }
37 private void saveLastProjectName()
38 {
39     String programConfigFile = getProgramConfigPath() + "/projectInfo.ini";
40     CommonFile temFile = new CommonFile(programConfigFile);
41     if(temFile.isExist())
42     {
43         temFile.delete();
44     }
45     if (temFile.openWrite())
46     {
47         tagProjectItem item;
48         for (int i = 0; i < mProjectItemList.size(); i++)
49         {
50             item = mProjectItemList.get(i);
```

```

1         String strLine = StringFunction.format("%s,%s,%s,%s,%s\r\n", item.strProjectName,
2 item.strProjectFilePath, item.strLastTime, item.strCreateTime, item.strOperator);
3         temFile.write(strLine);
4     }
5     temFile.close();
6 }
7 }
8     private boolean misOpenProject = false;
9     private String mstrLastProjectPath = "";
10    private String mstrProjectName = "";
11    private String mstrProgramPath = "";
12    private ArrayList<tagProjectItem> mProjectItemList = new ArrayList<>();
13 }
14 public class PointLibraryActivityV2 extends CommonBaseFragmentActivity implements
15 PointCommonFragment.OnChangeListener{
16     public static ArrayList<Long> cSelectPointIdItems = null;
17     private ePointLibraryMode pointLibraryMode = ePointLibraryMode.MODE_NULL;
18     private StakePointPreviewFragment pointPreviewFragment = null;
19     private CommonFragmentAdapter mFragmentAdapter;
20     protected void onCreate(Bundle savedInstanceState) {
21         super.onCreate(savedInstanceState);
22         setContentView(R.layout.activity_point_library_v2);
23         int mode = getIntent().getIntExtra("PointLibraryMode", -1);
24         pointLibraryMode = ePointLibraryMode.swigToEnum(mode);
25         initClickListener(R.id.imageButton_Mode, new View.OnClickListener() {
26             @Override
27             public void onClick(View v) {
28                 ViewPager viewPager = findViewById(R.id.viewPager_Fragment);
29                 CommonV4Fragment commonV4Fragment =
30 mFragmentAdapter.getItem(viewPager.getCurrentItem());
31                 if (commonV4Fragment instanceof PointListCommonFragment) {
32                     boolean bDisplayList =
33 ((PointListCommonFragment)commonV4Fragment).isDisplayList();
34                     ((PointListCommonFragment)commonV4Fragment).setDisplayMode(!bDisplayList);
35                     ImageView imageButtonMode = findViewById(R.id.imageButton_Mode);
36 imageButtonMode.setImageResource(bDisplayList?R.drawable.icon_point_mode_list:R.drawable.ic
37 on_point_mode_grid);
38                 }
39             }
40         });
41         mFragmentAdapter = new CommonFragmentAdapter(getSupportFragmentManager());
42         switch (pointLibraryMode)
43         {
44             case MODE_NULL:
45                 mFragmentAdapter.addFragment(new PointLibraryFragment(this));
46                 break;
47             case MODE_POINT_STAKEOUT: {
48                 setTitle(getString(R.string.title_point_stakeout));
49                 boolean bReturnToSurveyMain =
50 getIntent().getBooleanExtra("ReturnToSurveyMain", false);

```

```
1 PointLibraryFragment pointFragment = new PointLibraryFragment(this);
2 pointFragment.setStakeoutReturnToMain(bReturnToSurveyMain);
3 PointLibraryStakePointFragment stakePointFragment = new
4 PointLibraryStakePointFragment(this);
5 stakePointFragment.setStakeoutReturnToMain(bReturnToSurveyMain);
6 mFragmentManager.addFragment(pointFragment);
7 mFragmentManager.addFragment(stakePointFragment);
8 pointPreviewFragment = new StakePointPreviewFragment();
9 mFragmentManager.addFragment(pointPreviewFragment);
10 break;
11 }
12 case MODE_SELECT_POINT: {
13     setTitle(getString(R.string.title_coordinate_select));
14     PointLibraryFragment pointFragment = new PointLibraryFragment(this);
15     pointFragment.setSimpleSelect(true);
16     mFragmentManager.addFragment(pointFragment);
17     mFragmentManager.addFragment(new PointLibraryInputFragment(this));
18     break;
19 }
20 case MODE_SELECT_INPUT_POINT: {
21     setTitle(getString(R.string.title_coordinate_select));
22     PointLibraryFragment pointFragment = new PointLibraryFragment(this);
23     pointFragment.setSimpleSelect(true);
24     mFragmentManager.addFragment(pointFragment);
25     mFragmentManager.addFragment(new PointLibraryInputFragment(this));
26     break;
27 }
28 case MODE_SELECT_STAKE_POINT_LIST: {
29     setTitle(getString(R.string.title_coordinate_batch_select));
30     mFragmentManager.addFragment(new PointLibrarySelectStakeListFragment(this));
31     break;
32 }
33 case MODE_SELECT_POINT_LIST: {
34     setTitle(getString(R.string.title_coordinate_batch_select));
35     mFragmentManager.addFragment(new PointLibrarySelectListFragment(this));
36     break;
37 }
38 case MODE_SELECT_RESTORE_DELETE_POINT: {
39     setTitle(getString(R.string.title_restore_data));
40     mFragmentManager.addFragment(new PointLibraryRestoreDeletePointFragment(this));
41     break;
42 }
43 default: {
44     mFragmentManager.addFragment(new PointLibraryFragment(this));
45     break;
46 }
47 }
48 ViewPager viewPager = findViewById(R.id.viewPager_Fragment);
49 viewPager.setAdapter(mFragmentManager);
50 TabLayout tabLayout = findViewById(R.id.tabLayout_Fragment);
```

```

1      tabLayout.setupWithViewPager(viewPager);
2      if (mFragmentAdapter.getCount() <= 1)
3          tabLayout.setVisibility(View.GONE);
4      switch (pointLibraryMode)
5      {
6          case MODE_POINT_STAKEOUT:
7          case MODE_SELECT_INPUT_POINT:
8              viewPager.setCurrentItem(1);
9              break;
10     }
11     {
12         viewPager.addOnPageChangeListener(new ViewPager.OnPageChangeListener() {
13             @Override
14             public void onPageScrolled(int i, float v, int i1) {
15                 NoScrollViewPager viewPager = findViewById(R.id.viewPager_Fragment);
16                 CommonV4Fragment commonV4Fragment =
17 mFragmentAdapter.getItem(viewPager.getCurrentItem());
18                 if (pointPreviewFragment != null) {
19                     viewPager.setNoScroll(commonV4Fragment == pointPreviewFragment);
20                     if (commonV4Fragment == pointPreviewFragment) {
21
22 pointPreviewFragment.setPointItems(PointLibraryManage.getInstance().getStakePointList());
23                     }
24                 }
25                 if (commonV4Fragment instanceof PointListCommonFragment)
26                 {
27                     setVisibility(R.id.linearLayout_DisplayMode, View.VISIBLE);
28                 }
29                 else {
30                     setVisibility(R.id.linearLayout_DisplayMode, View.GONE);
31                 }
32                 commonV4Fragment.updateValueToUI();
33             }
34             @Override
35             public void onPageSelected(int i) {
36             }
37             @Override
38             public void onPageScrollStateChanged(int i) {
39             }
40         });
41     }
42 }
43 @Override
44 public void finish()
45 {
46     ViewPager viewPager = findViewById(R.id.viewPager_Fragment);
47     CommonV4Fragment commonV4Fragment =
48 mFragmentAdapter.getItem(viewPager.getCurrentItem());
49     if (commonV4Fragment instanceof PointCommonFragment) {
50         if

```

```
1 mFragmentAdapter.getItem(viewPager.getCurrentItem())).cancelBatchSelectionMode())
2     return;
3 }
4     super.finish();
5 }
6     @Override
7     public void onActivityResult(int requestCode, int resultCode, Intent data) {
8         ViewPager viewPager = findViewById(R.id.viewPager_Fragment);
9 mFragmentAdapter.getItem(viewPager.getCurrentItem()).onActivityResult(requestCode&0xFFFF,
10 resultCode, data);
11     }
12     @Override
13     public void onChanged() {
14         ViewPager viewPager = findViewById(R.id.viewPager_Fragment);
15         for (int i=0; i<mFragmentAdapter.getCount(); i++)
16             {
17                 if (i!= viewPager.getCurrentItem())
18                     {
19                         CommonV4Fragment commonV4Fragment = mFragmentAdapter.getItem(i);
20                         if (commonV4Fragment instanceof PointCommonFragment) {
21                             ((PointCommonFragment)commonV4Fragment).setDataReLoad();
22                         }
23                     }
24             }
25     }
26     @Override
27     public void importFileFinish(boolean success) {
28         mHandler.sendMessage(success?0:1);
29     }
30     @Override
31     public void showWaitingDialog(boolean isShow)
32     {
33         mHandler.sendMessage(isShow?5:6);
34     }
35     @Override
36     public void promptRepetitionName() {
37         mHandler.sendMessage(8);
38     }
39     @Override
40     public void searchFinish() {
41         mHandler.sendMessage(7);
42     }
43     @SuppressWarnings("HandlerLeak")
44     Handler mHandler = new Handler()
45     {
46         public void handleMessage(Message msg)
47         {
48             switch (msg.what)
49             {
50                 case 0: {
```

```

1      makeTextString(R.string.string_prompt_import_file_succeed);
2      setVisibility(R.id.waittingLayout, View.GONE);
3      ViewPager viewPager = findViewById(R.id.viewPager_Fragment);
4      CommonV4Fragment          commonV4Fragment          =
5  mFragmentManager.getItem(viewPager.getCurrentItem());
6      if (commonV4Fragment instanceof PointCommonFragment)
7          ((PointCommonFragment)commonV4Fragment).setDataReload();
8      commonV4Fragment.updateValueToUI();
9      onChanged();
10     }
11     break;
12     case 1:
13         makeTextString(R.string.string_prompt_import_file_failed);
14         setVisibility(R.id.waittingLayout, View.GONE);
15         break;
16     case 2: {
17         makeTextString(R.string.string_prompt_export_file_succeed);
18         setVisibility(R.id.waittingLayout, View.GONE);
19     }
20     break;
21     case 3:
22         makeTextString(R.string.string_prompt_export_file_failed);
23         setVisibility(R.id.waittingLayout, View.GONE);
24         break;
25     case 5:
26         setVisibility(R.id.waittingLayout, View.VISIBLE);
27         break;
28     case 6:
29         setVisibility(R.id.waittingLayout, View.GONE);
30         break;
31     case 7:
32     {
33         ViewPager viewPager = findViewById(R.id.viewPager_Fragment);
34         CommonV4Fragment          commonV4Fragment          =
35  mFragmentManager.getItem(viewPager.getCurrentItem());
36         commonV4Fragment.updateValueToUI();
37         setVisibility(R.id.waittingLayout, View.GONE);
38     }
39     break;
40     case 8:
41     {
42         setVisibility(R.id.waittingLayout, View.GONE);
43         new CustomDialog(AppFinalUtil.activityCurrent, R.string.string_prompt
44             , R.string.string_prompt_point_name_repetition_prompt
45             , R.string.button_continue, R.string.button_cancel)
46             .setDialogListener(new CustomDialog.OnDialogListener() {
47                 @Override
48                 public void onPositiveClick(View customView, DialogInterface
49  dialogInterface, int which) {
50                     ViewPager viewPager = findViewById(R.id.viewPager_Fragment);

```

```
1          CommonV4Fragment          commonV4Fragment          =
2  mFragmentManager.getItem(viewPager.getCurrentItem());
3          if (commonV4Fragment instanceof PointCommonFragment)
4              ((PointCommonFragment)commonV4Fragment).importPointFinish();
5          }
6          @Override
7          public void onNegativeClick(View customView, DialogInterface
8  dialogInterface, int which) {
9              }
10             }).showDialog();
11         }
12         break;
13         default:
14             break;
15     }
16 }
17 };
18 }
19 public class PointLibraryFragment extends PointListCommonFragment {
20     private boolean mbStakeoutEnable = false;
21     private boolean mbSimpleSelect = false;
22     private boolean mbReturnToSurveyMain = false;
23     public PointLibraryFragment(OnChangeListner listener) {
24         super(listener);
25     }
26     public void setStakeoutReturnToMain(boolean bReturnToSurveyMain)
27     {
28         mbStakeoutEnable = true;
29         mbReturnToSurveyMain = bReturnToSurveyMain;
30     }
31     public void setSimpleSelect(boolean bSimple)
32     {
33         mbSimpleSelect = bSimple;
34     }
35     @Override
36     public String getTitle() {
37         return AppFinalUtil.getString(R.string.title_library_coordinate_point);
38     }
39     protected void initCustom()
40     {
41         super.initCustom();
42         ((CustomGridPointItemAdapter)mGridItemListAdapter).setStakeoutMode(mbStakeoutEnable);
43         ((CustomGridPointItemAdapter)mGridItemListAdapter).setInvertedOrder(true);
44         mGridItemListAdapter.setEnabledBatchSelect(!mbReturnToSurveyMain);
45         mGridItemListAdapter.setItemButtonVisible(!mbReturnToSurveyMain);
46     }
47     @Override
48     protected void loadSearchData(ePointQueryType type, String strFind)
49     {
50         mAllPointListArray.clear();
```

```
1 mAllPointListArray.addAll(ProjectDbFileManage.getInstance().queryFindPointDataIds(type,
2 strFind));
3 }
4 @Override
5 public void onClickItem(int position)
6 {
7     if (mbSimpleSelect)
8     {
9         if (position >= 0) {
10             long pointId = mAllPointListArray.get(position);
11             Intent intent = new Intent();
12             intent.putExtra("ObjectID", pointId);
13             getActivity().setResult(AppFinalUtil.RETURN_CODE_VALUE_BACK, intent);
14             getActivity().finish();
15         }
16     }
17     else {
18         if (!mGridItemListAdapter.isBatchSelectionMode() &&
19             !mbReturnToSurveyMain && !mbStakeoutEnable &&
20             mGridItemListAdapter.getSelectedPosition() <= 0)
21         {
22             onClickEdit(position);
23         }
24         else {
25             super.onClickItem(position);
26         }
27     }
28 }
29 @Override
30 public void onClickApply() {
31     if (mbStakeoutEnable) {
32         int selectIndex = mGridItemListAdapter.getSelectedPosition();
33         if (ConfigStakeoutPoint.getInstance().isAutoStakeout()) {
34             ConfigStakeoutPoint.getInstance().setAutoStakeout(false);
35             ConfigStakeoutPoint.getInstance().saveConfig();
36         }
37
38         long pointId = mAllPointListArray.get(selectIndex);
39         PointLibraryManage.getInstance().setStakeEntity(pointId);
40         if (!mbReturnToSurveyMain) {
41             Intent intent = new Intent(getActivity(), MainPointSurveyActivity.class);
42             intent.putExtra(MainPointSurveyActivity.strSurveyWordModeId,
43 eSurveyWorkMode.WORK_MODE_STAKEOUT_POINT.swigValue());
44             startActivity(intent);
45         }
46         getActivity().finish();
47     }
48 }
49 @Override
50 public int getHeaderLayout() {
```

```
1     return R.layout.header_project_data_view;
2 }
3 @Override
4 public ArrayList<TextView> getHeadTextViews(View convertView) {
5     ArrayList<TextView> textViews = new ArrayList<>();
6     TextView temTextView = convertView.findViewById(R.id.textView_point_type);
7     textViews.add(temTextView);
8     temTextView = convertView.findViewById(R.id.textView_Name);
9     textViews.add(temTextView);
10    if (ConfigSystem.getInstance().isDisplayNorthFirst()) {
11        temTextView = convertView.findViewById(R.id.textView_North);
12        textViews.add(temTextView);
13        temTextView = convertView.findViewById(R.id.textView_East);
14        textViews.add(temTextView);
15    } else {
16        temTextView = convertView.findViewById(R.id.textView_East);
17        textViews.add(temTextView);
18        temTextView = convertView.findViewById(R.id.textView_North);
19        textViews.add(temTextView);
20    }
21    temTextView = convertView.findViewById(R.id.textView_Height);
22    textViews.add(temTextView);
23    temTextView = convertView.findViewById(R.id.textView_Latitude);
24    textViews.add(temTextView);
25    temTextView = convertView.findViewById(R.id.textView_Longitude);
26    textViews.add(temTextView);
27    temTextView = convertView.findViewById(R.id.textView_Altitude);
28    textViews.add(temTextView);
29    temTextView = convertView.findViewById(R.id.textView_Code);
30    textViews.add(temTextView);
31    temTextView = convertView.findViewById(R.id.textView_Mileage);
32    textViews.add(temTextView);
33    temTextView = convertView.findViewById(R.id.textView_Offset);
34    textViews.add(temTextView);
35    temTextView = convertView.findViewById(R.id.textView_BaseDistance);
36    textViews.add(temTextView);
37    temTextView = convertView.findViewById(R.id.textView_DateTime);
38    textViews.add(temTextView);
39    return textViews;
40 }
41 @Override
42 public int getDataSize() {
43     return mAllPointListArray.size();
44 }
45 @Override
46 public ArrayList<String> getDataByIndex(int index) {
47     if (index < 0 || index >= getDataSize())
48         return new ArrayList<>();
49     tagPointNode                pointNode                =
50 ProjectDbFileManage.getInstance().queryPointNode(mAllPointListArray.get(getDataSize())
```

```
1 index - 1), true);
2     ArrayList<String> arrayList = new ArrayList<>();
3     arrayList.add(pointNode.getPointType().swigValue() + "");
4     arrayList.add(pointNode.strPointName);
5     eLengthUnitType unitType = ConfigSystem.getInstance().getLengthUnitType();
6     eAngleFormatType angleFormatType = ConfigSystem.getInstance().getAngleFormatType();
7     tagBLHCoord blhCoord = pointNode.getBLHCoord();
8     tagNEhCoord nehCoord = pointNode.getNehCoord();
9     if (ConfigSystem.getInstance().isDisplayNorthFirst()) {
10         arrayList.add(StringFunction.valueOf(unitType.getLength(nehCoord.getNorth())));
11         arrayList.add(StringFunction.valueOf(unitType.getLength(nehCoord.getEast())));
12     } else {
13         arrayList.add(StringFunction.valueOf(unitType.getLength(nehCoord.getEast())));
14         arrayList.add(StringFunction.valueOf(unitType.getLength(nehCoord.getNorth())));
15     }
16     arrayList.add(StringFunction.valueOf(unitType.getLength(nehCoord.getHeight())));
17     arrayList.add(angleFormatType.toString(blhCoord.getLatitude(),
18 eAngleFormatType.BLMODE_B));
19     arrayList.add(angleFormatType.toString(blhCoord.getLongitude(),
20 eAngleFormatType.BLMODE_L));
21     arrayList.add(StringFunction.valueOf(unitType.getLength(blhCoord.getAltitude())));
22     arrayList.add(pointNode.strPointCode);
23     if (pointNode.stakeoutData != null && pointNode.stakeoutData.dataType.swigValue() >= 10)
24     {
25         eMileageType mileageType = ConfigSystem.getInstance().getMileageType();
26         arrayList.add(mileageType.toString(pointNode.stakeoutData.dMileage));
27     arrayList.add(StringFunction.valueOf(unitType.getLength(pointNode.stakeoutData.dOffset)));
28     }
29     else {
30         arrayList.add("");
31         arrayList.add("");
32     }
33     if (pointNode.getPointType().isSurveyPoint()) {
34     arrayList.add(StringFunction.valueOf(unitType.getLength(pointNode.gnssPositionData.getDistanceToBase())));
35     }
36     else{
37         arrayList.add("");
38     }
39     arrayList.add(pointNode.getDateTime().toString());
40     return arrayList;
41 }
42 }
43 }
44 public class SurveyPointEditActivity extends CommonBaseActivity
45 implements OnClickListener {
46     public static tagPointNode cEditPointNode = null;
47     byte[] bPhotoSketch = null;
48     RoverAntennaParam antenna = new RoverAntennaParam();
49     @Override
50     protected void onCreate(Bundle savedInstanceState)
```

```

1      {
2          super.onCreate(savedInstanceState);
3          setContentView(R.layout.activity_survey_point_edit);
4          if (cEditPointNode == null || cEditPointNode.gnssPositionData == null) {
5              finish();
6              return;
7          }
8          temAntenna.setType(cEditPointNode.gnssPositionData.getAntennaMeasureType());
9          temAntenna.setMeasureHeight(cEditPointNode.gnssPositionData.getAntennaMeasureHeight());
10         temAntenna.setAntennaInfo(cEditPointNode.gnssPositionData.getAntennaInfoData());
11         bPhotoSketch = ProjectDbFileManager.getInstance().getImageRemark(cEditPointNode.keyId);
12         if (bPhotoSketch != null)
13         {
14             Button eventImage = findViewById(R.id.button_ImageNote);
15             eventImage.setText(StringFunction.addRedPoint(eventImage.getText().toString()));
16         }
17         initialUI();
18         setTextViewFocus(R.id.editText_Name);
19     }
20     protected void initialUI()
21     {
22         setTextViewText(R.id.editText_Name, cEditPointNode.strPointName);
23         setTextViewText(R.id.editText_Code, cEditPointNode.strPointCode);
24         initClickListener(R.id.imageButton_CodeSelect, this);
25         initClickListener(R.id.button_ImageNote, this);
26         initClickListener(R.id.linearLayout_Antenna, this);
27         initClickListener(R.id.button_OK, this);
28         eLengthUnitType unitType = ConfigSystem.getInstance().getLengthUnitType();
29         String strAntennaHeight =
30         StringFunction.format("%s+%s%s", StringFunction.valueOf(unitType.getLength(temAntenna.getMeasureHeight()), true),
31         StringFunction.valueOf(unitType.getLength(temAntenna.getAntennaHeight()-temAntenna.getMeasureHeight()), unitType.getUnit()));
32         setTextViewText(R.id.textView_AntennaValue, strAntennaHeight);
33         if (cEditPointNode.gnssPositionData.getSensorType().swigValue() >=
34         eSensorType.TYPE_SENSOR_INCLINE.swigValue()) {
35             setEnabled(R.id.linearLayout_Antenna, false);
36         }
37         displayGnssInfo();
38     }
39     @Override
40     public void onClick(View arg0) {
41         if(arg0.getId() ==R.id.button_OK)
42         {
43             onButtonOK();
44         }else if(arg0.getId() == R.id.button_ImageNote)
45         {
46             Intent intent = new Intent(this, PhotoSketchActivity.class);
47             PhotoSketchActivity.bPhotoSketch = bPhotoSketch;
48             PhotoSketchActivity.cPointNode = cEditPointNode;

```

```
1      startActivityForResult(intent, R.id. button_ImageNote);
2  }
3  else if (R.id. imageButton_CodeSelect == arg0.getId()) {
4  DisplayMenuManage. getInstance(). doFunctionForId(eFunctionType. FUNCTION_TYPE_CODE_LIBRARY. sw
5  igValue());
6  }else if (R.id. LinearLayout_Antenna == arg0.getId()) {
7      Intent intent = new Intent();
8      intent.putExtra("AntennaMeasureType", temAntenna.getType().swigValue());
9      intent.putExtra("AntennaMeasureHeight", temAntenna.getMeasureHeight());
10     intent.putExtra("AntennaInfo", temAntenna.getAntennaInfo().toString());
11     intent.setClass(this, SettingRoverAntennaActivity.class);
12     startActivityForResult(intent, R.id. LinearLayout_Antenna);
13 }
14 }
15 private void onButtonOK() {
16     boolean bChangeValue = false;
17     String strPointName = getTextViewString(R.id. editText_Name);
18     String strPointCode = getTextViewString(R.id. editText_Code);
19     if(!strPointName.equals(cEditPointNode. strPointName) ||
20         !strPointCode.equals(cEditPointNode. strPointCode))
21     {
22         bChangeValue = true;
23         cEditPointNode. strPointName = strPointName;
24         cEditPointNode. strPointCode = strPointCode;
25     }
26     if (cEditPointNode. gnssPositionData.getAntennaMeasureType() != temAntenna.getType() ||
27         Math. abs(cEditPointNode. gnssPositionData.getAntennaMeasureHeight() -
28 temAntenna.getMeasureHeight()) > 1E-4 ||
29         !cEditPointNode. gnssPositionData.getAntennaInfoData(). toString(). equals(temAnte
30 nna.getAntennaInfo(). toString()))
31     {
32         bChangeValue = true;
33         cEditPointNode. gnssPositionData.setAntennaMeasureType(temAntenna.getType());
34 cEditPointNode. gnssPositionData.setAntennaMeasureHeight(temAntenna.getMeasureHeight());
35         cEditPointNode. gnssPositionData.setAntennaInfoData(temAntenna.getAntennaInfo());
36         cEditPointNode. gnssPositionData.updateAntennaHeight();
37     }
38     if (bChangeValue)
39     {
40         Intent intent = new Intent();
41         intent.putExtra("Position", getIntent().getIntExtra("Position", -1));
42         intent.putExtra("PointType", ePointType. POINT_TYPE_SURVEY. swigValue());
43         setResult(AppFinalUtil. RETURN_CODE_VALUE_BACK, intent);
44     }
45     finish();
46 }
47 @Override
48 public boolean onKeyDown(int keyCode, KeyEvent event) {
49     if (AppFinalUtil. isSurveyKey(keyCode)) {
50         onButtonOK();
```

```
1         return true;
2     }
3     ArrayList<eFunctionType> functionTypes = new ArrayList<>();
4     functionTypes.add(eFunctionType.FUNCTION_TYPE_ANTENNA_SETTING);
5     functionTypes.add(eFunctionType.FUNCTION_TYPE_CODE_LIBRARY);
6     eFunctionType functionType = ShortCutsManage.getInstance().getFunctionType(keyCode,
7 functionTypes);
8     if (functionType != eFunctionType.FUNCTION_TYPE_NULL) {
9         DisplayMenuManage.getInstance().doFunctionForId(functionType.swigValue());
10        return true;
11    }
12    return super.onKeyDown(keyCode, event);
13}
14@Override
15protected void onActivityResult(int requestCode, int resultCode, Intent data) {
16    if (R.id.linearlayout_antenna == requestCode)
17    {
18        if (data == null)
19            return;
20        temAntenna.setType(eAntennaMeasureType.swigToEnum(data.getIntExtra("AntennaMeasureType",
21 0)));
22        temAntenna.setMeasureHeight(data.getDoubleExtra("AntennaMeasureHeight", 0));
23        DeviceInfoAntenna infoAntenna = new DeviceInfoAntenna();
24        infoAntenna.parseString(data.getStringExtra("AntennaInfo"));
25        temAntenna.setAntennaInfo(infoAntenna);
26        displayGnssInfo();
27        eLengthUnitType unitType = ConfigSystem.getInstance().getLengthUnitType();
28        String strAntennaHeight =
29 StringFunction.format("%s+%s%s", StringFunction.valueOf(unitType.getLength(temAntenna.getMea
30 sureHeight()), true),
31 StringFunction.valueOf(unitType.getLength(temAntenna.getAntennaHeight()-temAntenna.getMeasu
32 reHeight()))), unitType.getUnit());
33        setTextViewText(R.id.textView_antennaValue, strAntennaHeight);
34        return;
35    }
36    super.onActivityResult(requestCode, resultCode, data);
37    if (R.id.button_imageNote == requestCode)
38    {
39        if(PhotoSketchActivity.IsModified) {
40            bPhotoSketch = PhotoSketchActivity.bPhotoSketch;
41            Button eventImage = findViewById(R.id.button_imageNote);
42            if (bPhotoSketch != null)
43            {
44                eventImage.setText(StringFunction.addRedPoint(getString(R.string.button_image_note)));
45            }
46            else {
47                eventImage.setText(getString(R.string.button_image_note));
48            }
49            ProjectDbFileManage.getInstance().updateImageRemark(cEditPointNode.keyId,
50 PhotoSketchActivity.bPhotoSketch);
```

```
1         PhotoSketchActivity. bPhotoSketch = null;
2     }
3 }
4 if(requestCode == R.id. imageButton_CodeSelect && data!=null)
5 {
6     String strCode = data.getStringExtra("resultValue");
7     if (strCode != null)
8     {
9         setContentView(R.id. editText_Code, strCode);
10    }
11 }
12 }
13 public void displayGnssInfo() {
14     GnssRecordPositionData temGnssData = cEditPointNode.gnssPositionData;
15     double dAddAntennaHeight = temAntenna.getAntennaHeight() - temGnssData.getPhaseHeight();
16     String strTemString = "";
17     strTemString = StringFunction. format("%s/%d", temGnssData.getSatInLock(),
18 temGnssData.getSatInView());
19     setContentView(R.id. editText_SolutionState, temGnssData.getStatusMark());
20     if (strTemString.length() > 0)
21         setContentView(R.id. editText_SatelliteNum, String. format("(%s)", strTemString));
22     else
23         setContentView(R.id. editText_SatelliteNum, strTemString);
24     eLengthUnitType unitType = ConfigSystem. getInstance().getLengthUnitType();
25     eAngleFormatType angleFormatType = ConfigSystem. getInstance().getAngleFormatType();
26     CustomTextViewListLayout linearLayoutCoordinate=
27 findViewById(R.id. linearLayout_Coordinate);
28     linearLayoutCoordinate.clear();
29     tagBLHCoord blhCoord = temGnssData.getWgs84BlhCoord();
30     tagNEhCoord temNehCoord = temGnssData.getNehCoord();
31     strTemString = angleFormatType.toString(blhCoord.getLatitude(),
32 eAngleFormatType. BLMODE_B);
33     if (ConfigSystem. getInstance().isDisplayNorthFirst()) {
34 linearLayoutCoordinate.add(getString(R.string. string_lat), strTemString, getString(R.string. s
35 tring_n),StringFunction. valueOf(unitType.getLength(temNehCoord.getNorth())));
36     }
37     else {
38 linearLayoutCoordinate.add(getString(R.string. string_lat), strTemString, getString(R.string. s
39 tring_e),StringFunction. valueOf(unitType.getLength(temNehCoord.getEast())));
40     }
41     strTemString = angleFormatType.toString(blhCoord.getLongitude(),
42 eAngleFormatType. BLMODE_L);
43     if (ConfigSystem. getInstance().isDisplayNorthFirst()) {
44 linearLayoutCoordinate.add(getString(R.string. string_lon), strTemString, getString(R.string. s
45 tring_e),StringFunction. valueOf(unitType.getLength(temNehCoord.getEast())));
46     }
47     else {
48 linearLayoutCoordinate.add(getString(R.string. string_lon), strTemString, getString(R.string. s
49 tring_n),StringFunction. valueOf(unitType.getLength(temNehCoord.getNorth())));
50     }
```

```
1  linearLayoutCoordinate.add(getString(R.string.string_alt), StringFunction.valueOf(unitType.g
2  etLength(blhCoord.getAltitude()-dAddAntennaHeight)), getString(R.string.string_h), StringFunc
3  tion.valueOf(unitType.getLength(temNehCoord.getHeight()-dAddAntennaHeight)));
4      if (cEditPointNode.gnssPositionData != null) {
5          boolean bVaild = true;
6          CCoordinateConvert coordinateConvert = new CCoordinateConvert();
7          tagRtcmCoordinateSystemParameter          rtcmCoordinateSystemParameter          =
8  cEditPointNode.gnssPositionData.getRtcmCoordinateSystemParameter();
9          if (rtcmCoordinateSystemParameter != null)
10         {
11             coordinateConvert.setDestEllipsoidParameter(rtcmCoordinateSystemParameter.getTargetEllipsoi
12             dParam());
13             coordinateConvert.setProjectionParameter(rtcmCoordinateSystemParameter.getProjectParam());
14         }
15         else {
16             tagCoordinateSystemParameter          systemParameter          =
17  cEditPointNode.gnssPositionData.getCoordinateSystemParameter();
18             if (systemParameter != null) {
19
20             coordinateConvert.setDestEllipsoidParameter(systemParameter.getEllipsoidParam());
21             coordinateConvert.setProjectionParameter(systemParameter.getProjectParam());
22         }
23         else {
24             bVaild = false;
25         }
26     }
27     if (bVaild)
28     {
29         double dDistance = CommonFunction.GetDistance(blhCoord.getLatitude() - 1E-6,
30 blhCoord.getLongitude() - 1E-6, blhCoord.getAltitude(),
31 blhCoord.getLatitude() + 1E-6, blhCoord.getLongitude() + 1E-6,
32 blhCoord.getAltitude());
33         tagBLHCoord temBlhCoord = new tagBLHCoord();
34         temBlhCoord.setLatitude(blhCoord.getLatitude() - 1E-6);
35         temBlhCoord.setLongitude(blhCoord.getLongitude() - 1E-6);
36         temBlhCoord.setAltitude(blhCoord.getAltitude());
37         tagNEhCoord tagNEhCoord1 = new tagNEhCoord();
38         coordinateConvert.BLHtoNEh(temBlhCoord, tagNEhCoord1);
39         temBlhCoord.setLatitude(blhCoord.getLatitude() + 1E-6);
40         temBlhCoord.setLongitude(blhCoord.getLongitude() + 1E-6);
41         tagNEhCoord tagNEhCoord2 = new tagNEhCoord();
42         coordinateConvert.BLHtoNEh(temBlhCoord, tagNEhCoord2);
43         double K = CommonFunction.GetDistance(tagNEhCoord1, tagNEhCoord2) / dDistance;
44         linearLayoutCoordinate.add(getString(R.string.label_scale_correction_factor), StringFunction
45         .valueOf(K, 10));
46     }
47 }
48 BaseChangeCorrectParameter correctParameter = temGnssData.getBaseCorrectParameter();
49 if (correctParameter != null)
50 {
```

```
1      tagNEhCoord nehCoord = correctParameter.getCorrectNehCoordParameter();
2      if (ConfigSystem.getInstance().isDisplayNorthFirst()) {
3          linearLayoutCoordinate.add(getString(R.string.label_adjustment_value_north),
4      StringFunction.valueOf(unitType.getLength(nehCoord.getNorth())));
5          linearLayoutCoordinate.add(getString(R.string.label_adjustment_value_east),
6      StringFunction.valueOf(unitType.getLength(nehCoord.getEast())));
7      }
8      else
9      {
10         linearLayoutCoordinate.add(getString(R.string.label_adjustment_value_east),
11     StringFunction.valueOf(unitType.getLength(nehCoord.getEast())));
12         linearLayoutCoordinate.add(getString(R.string.label_adjustment_value_north),
13     StringFunction.valueOf(unitType.getLength(nehCoord.getNorth())));
14     }
15     linearLayoutCoordinate.add(getString(R.string.label_adjustment_value_height),
16     StringFunction.valueOf(unitType.getLength(nehCoord.getHeight())));
17     }
18     CustomTextViewListLayout linearLayoutPrecision=
19     findViewById(R.id.linearLayout_Precision);
20     linearLayoutPrecision.clear();
21     strTemString = StringFunction.valueOf(temGnssData.getVelocity() * 3.6);
22     linearLayoutPrecision.add(getString(R.string.label_point_detail_speed),
23     StringFunction.valueOf(Math.ceil(temGnssData.getAgeOfDiff()), true),
24     getString(R.string.label_point_detail_heading), strTemString);
25     strTemString = "NA";
26     if(temGnssData.getPdop() > 1E-4) {
27         strTemString = StringFunction.valueOf(temGnssData.getPdop());
28     }
29     String strTemString2 = "NA";
30     if(temGnssData.getHrms() > 1E-4) {
31         strTemString2 = StringFunction.valueOf(unitType.getLength(temGnssData.getHrms()));
32     }
33     linearLayoutPrecision.add(getString(R.string.label_point_detail_pdop), strTemString,
34     getString(R.string.label_point_detail_hrms), strTemString2);
35     strTemString = "NA";
36     if(temGnssData.getHdop() > 1E-4) {
37         strTemString = StringFunction.valueOf(temGnssData.getHdop());
38     }
39     strTemString2 = "NA";
40     if(temGnssData.getVrms() > 1E-4) {
41         strTemString2 = StringFunction.valueOf(unitType.getLength(temGnssData.getVrms()));
42     }
43     linearLayoutPrecision.add(getString(R.string.label_point_detail_hdop), strTemString,
44     getString(R.string.label_point_detail_vrms), strTemString2);
45     strTemString = "NA";
46     if(temGnssData.getVrms() > 1E-4) {
47         strTemString = StringFunction.valueOf(temGnssData.getVdop());
48     }
49     linearLayoutPrecision.add(getString(R.string.label_point_detail_vdop), strTemString,
50     getString(R.string.label_point_detail_diff_delay), StringFunction.valueOf(Math.ceil(temGnssD
```

```
1  ata.getAgeOfDiff()), true));
2      linearLayoutPrecision.add(getString(R.string.string_record_average_count),
3  String.valueOf(temGnssData.getEpochCount()),
4  getString(R.string.string_record_average_count), String.valueOf(temGnssData.getElevMask()));
5      CustomTextViewListLayout linearLayoutOther= findViewById(R.id.linearLayout_Other);
6      linearLayoutOther.clear();
7      tagGnssRefStationItem refStationItem = temGnssData.getRefStationData();
8      if (refStationItem != null) {
9          linearLayoutOther.add(getString(R.string.label_point_detail_base_id),
10 refStationItem.getBaseId(),
11 getString(R.string.label_point_detail_ref_distance), StringFunction.valueOf(unitType.getLength(temGnssData.getDistanceToBase())));
12 }
13 }
14 linearLayoutOther.add(getString(R.string.label_point_detail_utcTime),
15 temGnssData.getDateTime().toString());
16 linearLayoutOther.add(getString(R.string.label_point_detail_localTime),
17 temGnssData.getLocalDateTime().toString());
18 linearLayoutOther.add(getString(R.string.string_instrument_serial_no),
19 temGnssData.getDeviceSN());
20 }
21 }
22 public class SurveySettingActivity extends CommonBaseFragmentActivity implements
23 OnClickListener {
24     private SettingFragmentAdapter mFragmentAdapter;
25     @Override
26     protected void onCreate(Bundle savedInstanceState) {
27         super.onCreate(savedInstanceState);
28         setContentView(R.layout.activity_survey_setting);
29         initialUI();
30     }
31     private void initialUI() {
32         initClickListener(R.id.button_OK, this);
33         initClickListener(R.id.button_Default, this);
34         initClickListener(R.id.button_Backspace, this);
35         initClickListener(R.id.button_Clear, this);
36         eSurveyWorkMode = surveyWorkMode
37 eSurveyWorkMode.swigToEnum(getIntent().getIntExtra("SurveyWorkMode",
38 WordModeManage.getInstance().getWorkMode().swigValue()));
39         ePointRecordMode recordMode = ePointRecordMode.POINT_RECORD_MODE_SMOOTH;
40         if (surveyWorkMode == eSurveyWorkMode.WORK_MODE_SURVEY_CONTROL_POINT)
41         {
42             recordMode = ePointRecordMode.POINT_RECORD_MODE_CONTROL;
43         }
44         else if (surveyWorkMode == eSurveyWorkMode.WORK_MODE_SURVEY_AUTO_POINT)
45         {
46             recordMode = ePointRecordMode.POINT_RECORD_MODE_CONTINUUM;
47         }
48         else {
49             if (RecordPointBase.getInstance().getPointRecordMode() ==
50 ePointRecordMode.POINT_RECORD_MODE_BASE_SURVEY)
```

```
1      recordMode = ePointRecordMode.POINT_RECORD_MODE_BASE_SURVEY;  
2      else if (ConfigPointSurvey.getInstance().isEnabledPPK())  
3          recordMode = ePointRecordMode.POINT_RECORD_MODE_STOP_GO;  
4  }  
5  mFragmentManager = new SettingFragmentManager(getSupportFragmentManager(), this);  
6  Bundle args=new Bundle();  
7  args.putInt("mode", recordMode.swigValue());  
8  if (surveyWorkMode == eSurveyWorkMode.WORK_MODE_CAD_EDIT)  
9  {  
10     mFragmentManager.addFragment(new SettingCadEditFragment());  
11     mFragmentManager.addFragment(new SettingMenuDisplayFragment());  
12 }  
13 else {  
14     SettingRecordItemConfigFragment fragment = new SettingRecordItemConfigFragment();  
15     fragment.setArguments(args);  
16     mFragmentManager.addFragment(fragment);  
17     switch (surveyWorkMode) {  
18         case WORK_MODE_STAKEOUT_POINT:  
19             mFragmentManager.addFragment(new SettingPointStakeoutFragment());  
20             break;  
21         case WORK_MODE_STAKEOUT_OBJECT:  
22             mFragmentManager.addFragment(new SettingObjectStakeoutFragment());  
23             break;  
24     }  
25     if (surveyWorkMode != eSurveyWorkMode.WORK_MODE_SURVEY_TEXT &&  
26         surveyWorkMode != eSurveyWorkMode.WORK_MODE_SURVEY_CONTROL_POINT &&  
27         surveyWorkMode != eSurveyWorkMode.WORK_MODE_SURVEY_AUTO_POINT) {  
28         mFragmentManager.addFragment(new SettingInfoDisplayFragment());  
29         mFragmentManager.addFragment(new SettingMenuDisplayFragment());  
30     }  
31 }  
32 ViewPager viewPager = findViewById(R.id.viewPager_Fragment);  
33 viewPager.setAdapter(mFragmentManager);  
34 viewPager.addOnPageChangeListener(new ViewPager.OnPageChangeListener() {  
35     @Override  
36     public void onPageScrolled(int i, float v, int il) {  
37     }  
38     @Override  
39     public void onPageSelected(int i) {  
40         ViewPager viewPager = findViewById(R.id.viewPager_Fragment);  
41         if (mFragmentManager.getItem(viewPager.getCurrentItem()).isSupportClear())  
42         {  
43             setVisibility(R.id.button_Clear, View.VISIBLE);  
44         }  
45         else {  
46             setVisibility(R.id.button_Clear, View.GONE);  
47         }  
48         if (mFragmentManager.getItem(viewPager.getCurrentItem()).isSupportBackspace())  
49         {  
50             setVisibility(R.id.button_Backspace, View.VISIBLE);
```

```
1         }
2     else {
3        .setVisibility(R.id.button_Backspace, View.GONE);
4     }
5     }
6     @Override
7     public void onPageScrollStateChanged(int i) {
8     }
9 });
10    TabLayout tabLayout = findViewById(R.id.tabLayout_Fragment);
11    tabLayout.setupWithViewPager(viewPager);
12    if(mFragmentAdapter.getCount() <= 1)
13        tabLayout.setVisibility(View.GONE);
14    }
15    @Override
16    public void onClick(View arg0) {
17        switch (arg0.getId()) {
18            case R.id.button_Default:
19                OnUseDefaultConfig();
20                break;
21            case R.id.button_Clear:
22                OnClearConfig();
23                break;
24            case R.id.button_Backspace: {
25                ViewPager viewPager = findViewById(R.id.viewPager_Fragment);
26                mFragmentAdapter.getItem(viewPager.getCurrentItem()).onButtonBackspace();
27            }
28            break;
29            case R.id.button_OK: {
30                for (int i=0; i<mFragmentAdapter.getCount(); i++) {
31                    mFragmentAdapter.getItem(i).updateValueFormUI();
32                }
33                setResult(AppFinalUtil.RETURN_CODE_DEFAULT);
34                finish();
35            }
36            break;
37            default:
38                break;
39        }
40    }
41    private void OnUseDefaultConfig() {
42        new
43        CustomDialog(this, R.string.string_prompt, R.string.string_prompt_message_used_default_settin
44        g, R.string.button_ok, R.string.button_cancel)
45        .setDialogListener(new CustomDialog.OnDialogListener() {
46            @Override
47            public void onPositiveClick(View customView, DialogInterface dialogInterface,
48            int which) {
49                ViewPager viewPager = findViewById(R.id.viewPager_Fragment);
50                mFragmentAdapter.getItem(viewPager.getCurrentItem()).onButtonDefault();
```

```
1         }
2         @Override
3         public void onNegativeClick(View customView, DialogInterface dialogInterface,
4 int which) {
5         }
6         }).showDialog();
7     }
8     private void OnClearConfig() {
9         new
10 CustomDialog(this, R.string. string_prompt, R.string. string_prompt_message_delete_setting
11         , R.string. button_ok, R.string. button_cancel)
12         .setDialogListener(new CustomDialog.OnDialogListener() {
13             @Override
14             public void onPositiveClick(View customView, DialogInterface dialogInterface,
15 int which) {
16                 ViewPager viewPager = findViewById(R.id. viewPager_Fragment);
17                 mFragmentManager.getItem(viewPager.getCurrentItem()).onButtonClear();
18             }
19             @Override
20             public void onNegativeClick(View customView, DialogInterface dialogInterface,
21 int which) {
22                 }
23             }).showDialog();
24         }
25         @Override
26         public void onActivityResult(int requestCode, int resultCode, Intent data) {
27             ViewPager viewPager = findViewById(R.id. viewPager_Fragment);
28 mFragmentManager.getItem(viewPager.getCurrentItem()).onActivityResult(requestCode&0xFFFF,
29 resultCode, data);
30         }
31     }
32 public class SettingInfoDisplayFragment extends SettingFragment {
33     ArrayList<Integer> mSelectArrayList = new ArrayList<Integer>();
34     private DisplayConfig mDisplayConfig = null;
35     @Nullable
36     @Override
37     public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, Bundle
38 savedInstanceState) {
39         if (mInfoView != null)
40             return mInfoView;
41         mInfoView
42 inflater.inflate(R.layout. layout_fragment_display_info_select, container, false);
43         eSurveyWorkMode wordMode = WordModeManage. getInstance().getWorkMode();
44         mDisplayConfig = ConfigDisplayInfoManage. getInstance().getDisplayConfig(wordMode);
45         mSelectArrayList = mDisplayConfig.getSelectedDisplayList();
46         CustomItemListLayout
47 mInfoView.findViewById(R.id. linearlayout_ItemList);
48         itemListLayout.setColumnCount(AppFinalUtil. isLanguageZh()?3:2);
49         itemListLayout.setOnClickListener(new CustomListItemRow.OnRowClickListener() {
50             @Override
```

```
1     public void onClick(String strValue, int id) {
2         if(mSelectArrayList.size() >= 8) {
3             return;
4         }
5         eDisplayItemType fieldType = eDisplayItemType.swigToEnum(id);
6         mSelectArrayList.add(fieldType.swigValue());
7         updateValueToUI();
8     }
9     @Override
10    public void onClickDelete(String strValue, int id) {
11    }
12    });
13    updateValueToUI();
14    setCheckBoxChecked(R.id.checkButton_DisplayName,
15    ConfigSystem.getInstance().isDisplayMapName());
16    setCheckBoxChecked(R.id.checkButton_DisplayCode, ConfigSystem.getInstance().isDisplayMapC
17    ode());
18    setCheckBoxChecked(R.id.checkButton_DisplayHeight, ConfigSystem.getInstance().isDisplayMa
19    pHeight());
20    CustomTextViewLayoutSelect temLayoutSelect =
21    mInfoView.findViewById(R.id.linearLayout_BackgroundColor);
22    temLayoutSelect.clearItem();
23    temLayoutSelect.addItem(getString(R.string.string_color_white));
24    temLayoutSelect.addItem(getString(R.string.string_color_black));
25    temLayoutSelect.setSelectedId(ConfigSystem.getInstance().getBackgroundColor() ==
26    Color.BLACK ? 1:0);
27    return mInfoView;
28    }
29    @Override
30    public void onButtonDefault()
31    {
32        mSelectArrayList = mDisplayConfig.getDefaultDisplayItems();
33        updateValueToUI();
34    }
35    public boolean isSupportBackspace()
36    {
37        return true;
38    }
39    @Override
40    public void onButtonBackspace()
41    {
42        if (mSelectArrayList.size() > 0)
43            mSelectArrayList.remove(mSelectArrayList.size()-1);
44        updateValueToUI();
45    }
46    @Override
47    public String getTitle() {
48        return AppFinalUtil.getString(R.string.string_display_info);
49    }
50    @Override
```

```

1     public void updateValueToUI() {
2         if (mInfoView == null)
3             return;
4         CustomInfoView infoView = mInfoView.findViewById(R.id. custom_display_info_view);
5         infoView.requestLayout(mSelectArrayList);
6         infoView.invalidate();
7         CustomItemListLayout                itemListLayout                =
8     mInfoView.findViewById(R.id. LinearLayout_ItemList);
9         itemListLayout.clear();
10        ArrayList<Integer>                enableSelectList                =
11    mDisplayConfig.getDisplayItemsNoInList(mSelectArrayList);
12        for(int i=0;i<enableSelectList.size();i++){
13            eDisplayItemType fieldType = eDisplayItemType. swigToEnum(enableSelectList.get(i));
14            itemListLayout.add(fieldType.getTitle(), fieldType. swigValue());
15        }
16        itemListLayout.updateValue();
17    }
18    @Override
19    public void updateValueFormUI() {
20        if (mInfoView == null)
21            return;
22        if (mDisplayConfig != null) {
23            mDisplayConfig.setSelectDisplayList(mSelectArrayList);
24            ConfigDisplayInfoManage. getInstance().saveConfig();
25        }
26        ConfigSystem. getInstance().setDisplayMapName(getCheckBoxChecked(R.id. checkButton_Display
27    Name));
28        ConfigSystem. getInstance().setDisplayMapCode(getCheckBoxChecked(R.id. checkButton_Display
29    Code));
30        ConfigSystem. getInstance().setDisplayMapHeight(getCheckBoxChecked(R.id. checkButton_Displ
31    ayHeight));
32        CustomTextViewLayoutSelect                temLayoutSelect                =
33    mInfoView.findViewById(R.id. LinearLayout_BackgroundColor);
34        ConfigSystem. getInstance().setBackgroundColor(temLayoutSelect.getSelectedId()
35    1?Color. BLACK:Color. WHITE);
36        ConfigSystem. getInstance().saveConfig();
37    }
38 }
39 public class SettingPointStakeoutFragment extends SettingFragment{
40     @Nullable
41     @Override
42     public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, Bundle
43     savedInstanceState) {
44         if (mInfoView != null)
45             return mInfoView;
46         mInfoView = inflater.inflate(R.layout. layout_setting_stakeout_point, container, false);
47         eLengthUnitType unitType = ConfigSystem. getInstance().getLengthUnitType();
48         setCheckBoxChecked(R.id. checkButton_Stakeout_Direction,
49     ConfigStakeout. getInstance().isPromptStakeoutDirection());
50         CustomTextViewLayoutSelect                layoutSelect                =

```

```
1 mInfoView.findViewById(R.id. layoutSelect_Direction_Reference);
2     layoutSelect.clearItem();
3     ArrayList<eDirectionReference> directionReferences =
4 DeviceCommand. getInstance().getAvailableDirectionReferenceTypes();
5     for (int i=0; i< directionReferences.size(); i++){
6         eDirectionReference directionReference = directionReferences.get(i);
7         layoutSelect.addItem(directionReference.getTitle(), directionReference.swigValue());
8     }
9     layoutSelect.registerItemSelectedInterface(new
10 CustomTextViewLayoutSelect.OnItemSelectedCallback() {
11         @Override
12         public void onItemSelected(View view, String label, int id) {
13             int visibility = eDirectionReference. swigToEnum(id) ==
14 eDirectionReference. TYPE_REFERENCE_POINT?View. VISIBLE:View. GONE;
15             setVisibility(R.id. viewListLayout_ReferencePoint, visibility);
16             setVisibility(R.id. view_line_ReferencePoint, visibility);
17         }
18     });
19 layoutSelect.setSelectedId(ConfigStakeout. getInstance().getStakeoutReference().swigValue());
20 EntityPoint referencePoint = ConfigStakeout. getInstance().getReferencePoint();
21 CustomTextViewListLayout viewListLayoutReferencePoint =
22 mInfoView.findViewById(R.id. viewListLayout_ReferencePoint);
23 viewListLayoutReferencePoint.setName(referencePoint.strName);
24 viewListLayoutReferencePoint.clear();
25 if (ConfigSystem. getInstance().isDisplayNorthFirst())
26     viewListLayoutReferencePoint.add(StringFunction. format("%s:%s",
27 AppFinalUtil. getString(R.string. string_display_bar_north),
28 StringFunction. valueOf(unitType.getLength(referencePoint.dNorth))), "",
29 StringFunction. format("%s:%s", AppFinalUtil. getString(R.string. string_display_bar_east),
30 StringFunction. valueOf(unitType.getLength(referencePoint.dEast))), "");
31 else
32     viewListLayoutReferencePoint.add(StringFunction. format("%s:%s",
33 AppFinalUtil. getString(R.string. string_display_bar_east),
34 StringFunction. valueOf(unitType.getLength(referencePoint.dEast))), "",
35 StringFunction. format("%s:%s", AppFinalUtil. getString(R.string. string_display_bar_north),
36 StringFunction. valueOf(unitType.getLength(referencePoint.dNorth))), "");
37 viewListLayoutReferencePoint.setOnClickListener(new View.OnClickListener() {
38     @Override
39     public void onClick(View v) {
40         EntityPoint referencePoint = ConfigStakeout. getInstance().getReferencePoint();
41         Intent intent = new Intent();
42         intent.putExtra("VaildCoordType", 0x01);
43         intent.putExtra("CoordinateType", eCoordinateType. TYPE_COORD_NEH.swigValue());
44         intent.putExtra("InputName", true);
45         intent.putExtra("Name", referencePoint.strName);
46         intent.putExtra("North", referencePoint.dNorth);
47         intent.putExtra("East", referencePoint.dEast);
48         intent.putExtra("Height", referencePoint.dHeight);
49         intent.setClass(getContext(), CoordinateInputActivity.class);
50         startActivityForResult(intent, R.id. viewListLayout_ReferencePoint&0xFFFF);
```

```

1      }
2  });
3      CustomTextViewLayoutSelectEdit          layoutSelectEdit          =
4  mInfoView.findViewById(R.id. linearLayout_PromptRange);
5      String          strTitle          =          StringFunction. format("%s(%s)",
6  getString(R.string. string_stakeout_prompt_range), unitType.getUnit());
7      layoutSelectEdit.setTitle(strTitle);
8      layoutSelectEdit.addItems(new String[]{"0.5", "1", "1.5", "2", "3", "5", "10"});
9      setUnitDoubleValue(R.id. linearLayout_PromptRange,
10 ConfigStakeout. getInstance().getPromptRange());
11      layoutSelectEdit = mInfoView.findViewById(R.id. linearLayout_RecordRange);
12      strTitle          =          StringFunction. format("%s(%s)",
13  getString(R.string. string_stakeout_record_range), unitType.getUnit());
14      layoutSelectEdit.setTitle(strTitle);
15      layoutSelectEdit.addItems(new String[]{"0.02", "0.05", "0.1", "0.2", "0.5", "1", "10"});
16      setUnitDoubleValue(R.id. linearLayout_RecordRange,
17 ConfigStakeout. getInstance().getStakeoutRecordRange());
18      CustomCheckBoxButton          checkButton          =
19 mInfoView.findViewById(R.id. checkButton_AutoStakeCompass);
20      checkButton.setChecked(ConfigStakeout. getInstance().isAutoEnterStakeCompass());
21      checkButton = mInfoView.findViewById(R.id. checkButton_AutoZoom);
22      checkButton.setChecked(ConfigStakeout. getInstance().isAutoZoom());
23      checkButton = mInfoView.findViewById(R.id. checkButton_AutoStakeout);
24      checkButton.setChecked(ConfigStakeoutPoint. getInstance().isAutoStakeout());
25      checkButton = mInfoView.findViewById(R.id. checkButton_AutoMarkStakeout);
26      checkButton.setChecked(ConfigStakeoutPoint. getInstance().isAutoMarkStakeout());
27      checkButton.setVisibility(View. GONE);
28      return mInfoView;
29  }
30  @Override
31  public void onActivityResult(int requestCode, int resultCode, Intent data) {
32      super.onActivityResult(requestCode, resultCode, data);
33      if(resultCode == AppFinalUtil. RETURN_CODE_VALUE_BACK) {
34          if ((requestCode&0xFFFF) == (R.id. viewListLayout_ReferencePoint&0xFFFF)) {
35              EntityPoint referencePoint = ConfigStakeout. getInstance().getReferencePoint();
36              referencePoint.strName = data.getStringExtra("Name");
37              referencePoint.dNorth = data.getDoubleExtra("North", 0);
38              referencePoint.dEast = data.getDoubleExtra("East", 0);
39              referencePoint.dHeight = data.getDoubleExtra("Height", 0);
40              CustomTextViewListLayout          viewListLayoutReferencePoint          =
41  mInfoView.findViewById(R.id. viewListLayout_ReferencePoint);
42              viewListLayoutReferencePoint.setName(referencePoint.strName);
43              viewListLayoutReferencePoint.clear();
44              eLengthUnitType unitType = ConfigSystem. getInstance().getLengthUnitType();
45              if (ConfigSystem. getInstance().isDisplayNorthFirst())
46                  viewListLayoutReferencePoint.add(StringFunction. format("%s:%s",
47  AppFinalUtil. getString(R.string. string_display_bar_north),
48  StringFunction. valueOf(unitType.getLength(referencePoint.dNorth))),          "",
49  StringFunction. format("%s:%s",          AppFinalUtil. getString(R.string. string_display_bar_east),
50  StringFunction. valueOf(unitType.getLength(referencePoint.dEast))),          "");

```

```

1         else
2             viewListLayoutReferencePoint.add(StringFunction.format("%s:%s",
3 AppFinalUtil.getString(R.string.string_display_bar_east),
4 StringFunction.valueOf(unitType.getLength(referencePoint.dEast))), "",
5 StringFunction.format("%s:%s", AppFinalUtil.getString(R.string.string_display_bar_north),
6 StringFunction.valueOf(unitType.getLength(referencePoint.dNorth))), "");
7     }
8 }
9 }
10 @Override
11 public void onButtonDefault() {
12     CustomCheckBoxButton checkButton =
13 mInfoView.findViewById(R.id.checkButton_AutoStakeCompass);
14     checkButton.setChecked(false);
15     checkButton = mInfoView.findViewById(R.id.checkButton_AutoZoom);
16     checkButton.setChecked(false);
17     checkButton = mInfoView.findViewById(R.id.checkButton_AutoStakeout);
18     checkButton.setChecked(false);
19     checkButton = mInfoView.findViewById(R.id.checkButton_AutoMarkStakeout);
20     checkButton.setChecked(false);
21     eLengthUnitType unitType = ConfigSystem.getInstance().getLengthUnitType();
22     checkButton = mInfoView.findViewById(R.id.checkButton_Stakeout_Direction);
23     checkButton.setChecked(DeviceCommand.getInstance().isTotalStation());
24     CustomTextViewLayoutSelect layoutSelect =
25 mInfoView.findViewById(R.id.layoutSelect_Direction_Reference);
26     layoutSelect.setSelectedId(-1);
27     CustomTextViewLayoutSelectEdit layoutSelectEdit =
28 mInfoView.findViewById(R.id.linearLayout_PromptRange);
29     layoutSelectEdit.setText(StringFunction.valueOf(unitType.getLength(1), true));
30     layoutSelectEdit = mInfoView.findViewById(R.id.linearLayout_RecordRange);
31     layoutSelectEdit.setText(StringFunction.valueOf(unitType.getLength(0.02), true));
32 }
33 @Override
34 public String getTitle() {
35     return AppFinalUtil.getString(R.string.title_stakeout_setting);
36 }
37 @Override
38 public void updateValueToUI() {
39 }
40 @Override
41 public void updateValueFormUI() {
42     if (mInfoView == null)
43         return;
44     CustomCheckBoxButton checkButton =
45 mInfoView.findViewById(R.id.checkButton_AutoStakeCompass);
46     ConfigStakeout.getInstance().setAutoEnterStakeCompass(checkButton.isChecked());
47     checkButton = mInfoView.findViewById(R.id.checkButton_AutoZoom);
48     ConfigStakeout.getInstance().setAutoZoom(checkButton.isChecked());
49     checkButton = mInfoView.findViewById(R.id.checkButton_AutoStakeout);
50     ConfigStakeoutPoint.getInstance().setAutoStakeout(checkButton.isChecked());

```

```

1      checkButton = mInfoView.findViewById(R.id. checkButton_AutoMarkStakeout);
2      ConfigStakeoutPoint. getInstance(). setAutoMarkStakeout (checkButton. isChecked());
3      eLengthUnitType unitType = ConfigSystem. getInstance(). getLengthUnitType();
4      CustomTextViewLayoutSelectEdit          layoutSelectEdit          =
5 mInfoView.findViewById(R.id. linearLayout_PromptRange);
6 ConfigStakeout. getInstance(). setPromptRange (unitType. getMeterLength(layoutSelectEdit. getDoubleValue()));
7
8      layoutSelectEdit = mInfoView.findViewById(R.id. linearLayout_RecordRange);
9 ConfigStakeout. getInstance(). setStakeoutRecordRange (unitType. getMeterLength(layoutSelectEdit. getDoubleValue()));
10
11     checkButton = mInfoView.findViewById(R.id. checkButton_Stakeout_Direction);
12     ConfigStakeout. getInstance(). setPromptStakeoutDirection (checkButton. isChecked());
13     CustomTextViewLayoutSelect          layoutSelect          =
14 mInfoView.findViewById(R.id. layoutSelect_Direction_Reference);
15 ConfigStakeout. getInstance(). setStakeoutReference (eDirectionReference. swigToEnum(layoutSelect. getSelectedId()));
16
17     ConfigStakeoutPoint. getInstance(). saveConfig();
18     ConfigStakeout. getInstance(). saveConfig();
19 }
20 }
21 public class SettingMenuDisplayFragment extends SettingFragment {
22     private DisplayConfig mDisplayConfig = null;
23     private ArrayList<Integer> mSelectArrayList = new ArrayList<Integer>();
24     @Nullable
25     @Override
26     public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, Bundle savedInstanceState) {
27         if (mInfoView != null)
28             return mInfoView;
29         mInfoView
30 inflater.inflate(R.layout. layout_fragment_menu_info_select, container, false);
31         eSurveyWorkMode wordMode = WordModeManage. getInstance(). getWorkMode();
32         mDisplayConfig = ConfigDisplayInfoManage. getInstance(). getDisplayConfig(wordMode);
33         mSelectArrayList = mDisplayConfig. getSelectMenuList();
34         CustomItemListLayout          itemListLayout          =
35 mInfoView.findViewById(R.id. linearLayout_SelectList);
36         itemListLayout. setColumnCount(1);
37         itemListLayout. setOnClickListener(new CustomListItemRow. OnClickListener() {
38             @Override
39             public void onClick(String strValue, int id) {
40                 eFunctionType functionType = eFunctionType. swigToEnum(id);
41                 mSelectArrayList. add(functionType. swigValue());
42                 updateValueToUI();
43             }
44             @Override
45             public void onClickDelete(String strValue, int id) {
46             }
47         });
48         itemListLayout = mInfoView.findViewById(R.id. linearLayout_SelectedList);
49         itemListLayout. setColumnCount(1);
50

```

```
1      itemListLayout.setOnClickListener(new CustomListItemRow.OnRowClickListener() {
2          @Override
3          public void onClick(String strValue, int id) {
4              eFunctionType functionType = eFunctionType.swigToEnum(id);
5              mSelectArrayList.remove(Integer.valueOf(functionType.swigValue()));
6              updateValueToUI();
7          }
8          @Override
9          public void onClickDelete(String strValue, int id) {
10             }
11     });
12     return mInfoView;
13 }
14 @Override
15 public void onButtonDefault()
16 {
17     mSelectArrayList = mDisplayConfig.getDefaultMenuItems();
18     updateValueToUI();
19 }
20 public boolean isSupportClear()
21 {
22     return true;
23 }
24 @Override
25 public void onButtonClear()
26 {
27     mSelectArrayList.clear();
28     updateValueToUI();
29 }
30 @Override
31 public String getTitle() {
32     return AppFinalUtil.getString(R.string.string_display_menu);
33 }
34 @Override
35 public void updateValueToUI() {
36     if (mInfoView == null)
37         return;
38     CustomItemListLayout itemListLayout =
39 mInfoView.findViewById(R.id.linearLayout_SelectList);
40     itemListLayout.clear();
41     ArrayList<Integer> listAvailable =
42 mDisplayConfig.getMenuItemsNoInList(mSelectArrayList);
43     for(Integer id:listAvailable){
44         if (id == eFunctionType.FUNCTION_TYPE_ELECTRON_BUBBLE.swigValue() &&
45             DeviceCommandParse.getInstance().getSensorModel() != ESensorModel.TiltSurvey)
46             continue;
47         itemListLayout.add(eFunctionType.swigToEnum(id));
48     }
49     itemListLayout.updateValue();
50     itemListLayout = mInfoView.findViewById(R.id.linearLayout_SelectedList);
```

```
1     itemListLayout.clear();
2     for(int i = mSelectArrayList.size()-1; i>=0; i--){
3         itemListLayout.add(eFunctionType.swigToEnum(mSelectArrayList.get(i)));
4     }
5     itemListLayout.updateValue();
6 }
7 @Override
8 public void updateValueFormUI() {
9     if (mDisplayConfig != null) {
10         mDisplayConfig.setSelectMenuList(mSelectArrayList);
11         ConfigDisplayInfoManage.getInstance().saveConfig();
12     }
13 }
14 }
15 public class AboutProgramActivity extends CommonEventBaseActivity implements OnClickListener
16 {
17     private RegisterTransOutSocketClient transOutSocketClient = new
18 RegisterTransOutSocketClient();
19     @Override
20     public void onCreate(Bundle savedInstanceState)
21     {
22         super.onCreate(savedInstanceState);
23         setContentView(R.layout.activity_program_about);
24         if (AppFinalUtil.getAppID() == eAppIdentifierType.APP_ID_SURVEY_GINTEC)
25             setVisibility(R.id.imageView_Icon, View.VISIBLE);
26             setVisibility(R.id.button_Copy, View.GONE);
27             initialUI();
28             return;
29     }
30     private void initialUI()
31     {
32         String strVersion = StringFunction.format("V%s", AppFinalUtil.getAppVersion());
33         initClickListener(R.id.button_New_Version, this);
34         initClickListener(R.id.button_Activation, this);
35         initClickListener(R.id.button_Activation_Trans_out, this);
36         initClickListener(R.id.button_Feedback, this);
37         initClickListener(R.id.button_Apply, this);
38         initClickListener(R.id.textViewDeviceId, this);
39         String strCompany = AppFinalUtil.getAppID().getCompanyName();
40         ImageView temTextView = findViewById(R.id.imageView_logo);
41         try {
42             String strImageFileName = "";
43             strImageFileName = StringFunction.format("s/s",
44 ProjectManage.GetInstance().getProgramCachePath(), ".app_qr_code.png");
45             FileInputStream fis = new FileInputStream(strImageFileName);
46             temTextView.setImageBitmap(BitmapFactory.decodeStream(fis));
47         } catch (Exception e) {
48             temTextView.setImageResource(AppFinalUtil.getAppID().getQRCodeId());
49         }
50         setTextViewText(R.id.textView_Version, strVersion);
```

```

1      setContentView(R.id. textView_Company, strCompany);
2      setContentView(R.id. textView_Copyright, AppFinalUtil. getAppID(). getCopyright());
3      if(VersionDownloadClient. isUpdateAvailable()) {
4          Button eventAd = findViewById(R.id. button_New_Version);
5          setContentView(R.id. button_New_Version, getString(R.string. label_new_version));
6          eventAd. setText(StringFunction. addRedPoint(eventAd. getText(). toString()));
7      }
8      updateActivationInfo();
9  }
10 @Override
11 public void finish()
12 {
13     transOutSocketClient. disconnect();
14     super. finish();
15 }
16 private void updateActivationInfo() {
17     int                                     overDateFunc                               =
18 RegisterManage. GetInstance(). isRegOverDate(eMainMenuType. MENU_TYPE_NULL);
19     boolean displayPVersion = false;
20     if (AppFinalUtil. getAppID(). isChinaVersion() &&
21         (RegisterManage. GetInstance(). getValidFunction()&0x02) <= 0)
22     {
23         if (AppFinalUtil. isLanguageZhCh())
24         {
25             if (CustomGnssData. GetInstance(). isValidate() &&
26                 CustomGnssData. GetInstance(). getSolutionType()           !=
27 eGnssSolutionType. FIX_TYPE_INVALID)
28             {
29                 boolean                                     bPointInChina           =
30 RegisterManage. GetInstance(). pointInChina(CustomGnssData. GetInstance(). getLatitude(),
31 CustomGnssData. GetInstance(). getLongitude());
32                 if (!bPointInChina)
33                 {
34                     displayPVersion = true;
35                 }
36             }
37         }
38     else {
39         displayPVersion = true;
40     }
41 }
42 if ((RegisterManage. GetInstance(). getChinaLimitCode()&0x01) <=0 &&
43     (RegisterManage. GetInstance(). getValidFunction()&0x01) <=0 &&
44     !displayPVersion)
45 {
46     setVisibility(R.id. labelDateTime, View. GONE);
47     setVisibility(R.id. textViewDateTime, View. GONE);
48     setVisibility(R.id. linearLayoutAuthorization, View. GONE);
49 }
50 else {

```

```

1      setVisibility(R.id. labelDateTime, View. VISIBLE);
2      setVisibility(R.id. textViewDateTime, View. VISIBLE);
3  }
4      setVisibility(R.id. LinearLayout_PVersion, displayPVersion?View. VISIBLE:View. GONE);
5      setTextViewText(R.id. textViewDeviceId, RegisterManage. GetInstance().getDeviceID());
6      if (displayPVersion)
7      {
8          int    expireDate    =    Math.min(RegisterManage. GetInstance().getExpireDate1(),
9 RegisterManage. GetInstance().getExpireDate2());
10         int nYear = expireDate/10000;
11         int nMonth = (expireDate%10000)/100;
12         int nDay = expireDate%100;
13         if(nYear<2000 || nMonth==0 || nDay==0)
14             setTextViewText(R.id. textViewPDateTime, getString(R.string. SYS_FS_NOT_ENABLED));
15         else {
16             String strDateTime = nYear + "-" + nMonth + "-" + nDay;
17             setTextViewText(R.id. textViewPDateTime, strDateTime);
18         }
19         if (expireDate - RegisterManage. GetInstance().getRealTimeDate() < 100 &&
20             RegisterManage. GetInstance().getExpireDate1() - expireDate > 0 &&
21             (overDateFunc&0x01) <= 0)
22             setVisibility(R.id. button_Apply, View. VISIBLE);
23         else
24             setVisibility(R.id. button_Apply, View. GONE);
25     }
26     int expireDate = RegisterManage. GetInstance().getExpireDate1();
27     if (!displayPVersion && ((overDateFunc&0x02) > 0 ||
28 (RegisterManage. GetInstance().getValidFunction()&0x02) > 0) &&
29         expireDate != RegisterManage. GetInstance().getExpireDate2())
30     {
31         if (AppFinalUtil. isLanguageZhCh())
32         {
33             if (CustomGnssData. getInstance().isValidate() &&
34                 CustomGnssData. getInstance().getSolutionType() !=
35 eGnssSolutionType. FIX_TYPE_INVALID)
36             {
37                 boolean    bPointInChina    =
38 RegisterManage. GetInstance().pointInChina(CustomGnssData. getInstance().getLatitude(),
39 CustomGnssData. getInstance().getLongitude());
40                 if (!bPointInChina)
41                 {
42                     expireDate    =    Math.min(expireDate,
43 RegisterManage. GetInstance().getExpireDate2());
44                 }
45             }
46         }
47         else {
48             expireDate = Math.min(expireDate, RegisterManage. GetInstance().getExpireDate2());
49         }
50     }

```

```
1      int nYear = expireDate/10000;
2      int nMonth = (expireDate%10000)/100;
3      int nDay = expireDate%100;
4      if(nYear<2000||nMonth==0||nDay==0) {
5          setContentView(R.id.textViewDateTime, getString(R.string.SYS_FS_NOT_ENABLED));
6          setVisibility(R.id.linearLayoutAuthorization, View.GONE);
7      }
8      else {
9          String strDateTime = nYear + "-" + nMonth + "-" + nDay;
10         String strRegisterKeyId = ConfigSystem.getInstance().getSoftwareRegisterKeyId();
11         if (strRegisterKeyId.length()>8)
12             {
13             if (AppFinalUtil.getAppID() == eAppIdentifierType.APP_ID_SURVEY_GINTEC ||
14                 AppFinalUtil.getAppID().isSouthVersion()) {
15                 setVisibility(R.id.linearLayoutAuthorization, View.VISIBLE);
16                 String strAuthorization = StringFunction.format("%s***%s",
17 strRegisterKeyId.substring(0, 2), strRegisterKeyId.substring(strRegisterKeyId.length() - 6));
18                 setContentView(R.id.textViewAuthorization, strAuthorization);
19             }
20             else {
21                 setVisibility(R.id.linearLayoutAuthorization, View.GONE);
22                 strDateTime += StringFunction.format("(%s***%s)", strRegisterKeyId.substring(0,
23 2), strRegisterKeyId.substring(strRegisterKeyId.length()-6));
24             }
25         }
26         else {
27             setVisibility(R.id.linearLayoutAuthorization, View.GONE);
28         }
29         setContentView(R.id.textViewDateTime, strDateTime);
30     }
31 }
32 @Override
33 public void onConfigurationChanged (Configuration newConfig) {
34     super.onConfigurationChanged(newConfig);
35     setContentView(R.layout.activity_program_about);
36     initialUI();
37 }
38 @Override
39 public void onClick(View arg0)
40 {
41     switch (arg0.getId()) {
42         case R.id.button_New_Version:
43             RegisterAutoManageClient.getInstance().checkVersionUpdate();
44             showWaitingDialog(true);
45             break;
46         case R.id.button_Apply:
47             {
48                 startActivityForResult(new Intent(this, ApplyForRegisterCodeActivity.class),
49 R.id.button_Apply);
50             }
51         }
```

```
1         break;
2     case R.id. button_Activation:
3         startActivityForResult(new Intent(this, SoftwareActivateActivity.class),
4 R.id. button_Activation);
5         break;
6     case R.id. button_Activation_Trans_out:
7         if (RegisterManage. GetInstance().getValidFunction() > 0)
8         {
9             new CustomDialog(this, getString(R.string. string_prompt),
10                getString(R.string. string_trans_out_register),
11                getString(R.string. button_ok), getString(R.string. button_cancel))
12                .setDialogListener(new CustomDialog. OnDialogListener() {
13                    @Override
14                public void onPositiveClick(View customView, DialogInterface
15 dialogInterface, int which) {
16                    showWaitingDialog(true);
17                    transOutSocketClient.transOutActivation();
18                }
19                @Override
20                public void onNegativeClick(View customView, DialogInterface
21 dialogInterface, int which) {
22                }
23                }).showDialog();
24        }
25        else {
26            makeTextString(getString(R.string. string_no_trans_out_register_info));
27        }
28        break;
29    case R.id. button_Feedback:
30        startActivity(new Intent(this, UserFeedbackActivity.class));
31        break;
32    case R.id. textViewDeviceId:
33        {
34            try {
35                MultiFormatWriter writer = new MultiFormatWriter();
36                String info = RegisterManage. GetInstance().getDeviceID();//输入的内容
37                String contents=new String(info.getBytes("UTF-8"), "ISO-8859-1");
38                BitMatrix matrix = writer.encode(contents, BarcodeFormat. CODE_128, 1000, 250);
39                BarcodeEncoder encoder = new BarcodeEncoder();
40                final Bitmap bitmap = encoder.createBitmap(matrix);
41                LinearLayout linearLayout = new LinearLayout(this);
42                linearLayout.setOrientation(LinearLayout. VERTICAL);
43                TextView textView = new TextView(this);
44                textView.setGravity(Gravity. CENTER_HORIZONTAL);
45                textView.setText(info);
46                textView.setTextSize(20);
47                ImageView image = new ImageView(this);
48                image.setAdjustViewBounds(true);
49                image.setImageBitmap(bitmap);
50                linearLayout.addView(image);
```

```
1      linearLayout.addView(textView);
2      new CustomDialog(this, linearLayout ,
3          getString(R.string.dialog_scan_code_128_title),
4          null, getString(R.string.button_ok))
5          .setDialogListener(new CustomDialog.OnDialogListener() {
6              @Override
7                  public void onPositiveClick(View customView, DialogInterface
8 dialogInterface, int which) {
9                  }
10             @Override
11                 public void onNegativeClick(View customView, DialogInterface
12 dialogInterface, int which) {
13                 }
14             }).showDialog();
15         } catch (WriterException e) {
16             e.printStackTrace();
17         } catch (Exception e) {
18         }
19     }
20     break;
21 }
22 }
23 @Override
24 protected void onActivityResult(int requestCode, int resultCode, Intent data) {
25     super.onActivityResult(requestCode, resultCode, data);
26     if (requestCode == R.id.button_Activation || requestCode == R.id.button_Apply)
27     {
28         updateActivationInfo();
29     }
30 }
31 public void onEventMainThread(MainEvent.NetworkVersionConnectStatus result) {
32     if (!result.isConnectSuccess())
33     {
34         makeTextString(getString(R.string.string_prompt_connect_server_overtime));
35     }
36     showWaitingDialog(false);
37 }
38 public void onEventMainThread(MainEvent.NetworkActivationUpdateStatus result) {
39     updateActivationInfo();
40 }
41 public void onEventMainThread(MainEvent.NetworkTransOutActivationErrorStatus result) {
42     if (result.getErrorCode() != eErrorCode.SUCCESS)
43     {
44         makeTextString(result.getErrorCode().getTitle());
45     }
46     showWaitingDialog(false);
47     updateActivationInfo();
48 }
49 }
```