Update emv configuration instructions

Sdk provides interface functions to add, delete, find and update operations
updateEmvAPP(EMVDataOperation operationType, ArrayList<String> appTag)
updateEmvCAPK(EMVDataOperation operationType, ArrayList<String> ridTag)
Take updateEmvAPP(EMVDataOperation operationType, ArrayList<String> appTag) as an example.

Note: There are pre-loaded emv configurations in the device, and the default preloading of ten groups. All emv operating ranges are within the preloaded number
Example: The device preloads ten groups. The current execution of the add operation fails. You can delete it and add it.

```
public enum EMVDataOperation {
    Clear, Add, Delete, AttainList, update, getEmv
}
```

Find:

AttainList // Get the list of AIDs in the device

Code example:

```
pos.updateEmvAPP(EMVDataOperation.AttainList, new ArrayList<String>());
//not recommended to pass null
```

listener function

```
@Override
public void onReturnGetEMVListResult(String arg0) {
   // TODO Auto-generated method stub
   TRACE.d("onReturnGetEMVListResult(String arg0):" + arg0);

   if (arg0 != null && arg0.length() > 0) {
      statusEditText.setText("The emv list is : " + arg0);
   }
}
```

getEmv // Get the parameters corresponding to the AID

Code example:

```
list.add(EmvAppTag.Application_Identifier_AID_terminal+"0000000000000
0000000000000000000");
pos.updateEmvAPP(EMVDataOperation.getEmv, list);
```

Note: no pass AID or device not searched for incoming corresponding AID returns failed

listener function

```
@Override
public void onReturnGetEMVListResult(String arg0) {
   // TODO Auto-generated method stub
   TRACE.d("onReturnGetEMVListResult(String arg0):" + arg0);

   if (arg0 != null && arg0.length() > 0) {
      statusEditText.setText("The emv list is : " + arg0);
   }
}
```

Delete:

Clear// Delete the list of preloaded AIDs in the device

Code example:

```
pos.updateEmvAPP(EMVDataOperation.Clear, new ArrayList<String>());
//not recommended to pass null
```

listener function

```
@Override
public void onReturnUpdateEMVResult(boolean arg0) {
   // TODO Auto-generated method stub
   TRACE.d("onReturnUpdateEMVResult(boolean arg0):" + arg0);

   if (arg0) {
      statusEditText.setText("update EMV app success");
   } else {
      statusEditText.setText("update emv app fail~");
      ConfigUtil.putReadXmlStatus(MainActivity.this, false);
   }
}
```

Delete // Delete the corresponding AID node

Code example:

```
list.add(EmvAppTag.Application_Identifier_AID_terminal+"0000000000000
0000000000000000000");
pos.updateEmvAPP(EMVDataOperation.Delete,list);
```

Note: no pass AID or device not searched for incoming corresponding AID returns failed

listener function

```
@Override
public void onReturnUpdateEMVResult(boolean arg0) {
   // TODO Auto-generated method stub
   TRACE.d("onReturnUpdateEMVResult(boolean arg0):" + arg0);

   if (arg0) {
      statusEditText.setText("update EMV app success");
   } else {
      statusEditText.setText("update emv app fail~");
      ConfigUtil.putReadXmlStatus(MainActivity.this, false);
   }
}
```

Update:

Update// Update the parameters corresponding to the AID

Code example:

```
list.add(EmvAppTag.Application_Identifier_AID_terminal+"0000000000000
0000000000000000000");
list.add(EmvAppTag.Transaction_Currency_Code+"0208");
pos.updateEmvAPP(EMVDataOperation.update,list);
```

Note: If the incoming parameter does not contain the AID, the corresponding incoming parameters of all AIDs in the device are updated.

The incoming parameter contains the AID, and the AID list in the device does not have an AID update failure.

listener function

```
@Override
public void onReturnUpdateEMVResult(boolean arg0) {
   // TODO Auto-generated method stub
   TRACE.d("onReturnUpdateEMVResult(boolean arg0):" + arg0);

   if (arg0) {
      statusEditText.setText("update EMV app success");
   } else {
      statusEditText.setText("update emv app fail~");
      ConfigUtil.putReadXmlStatus(MainActivity.this, false);
   }
}
```

Add:

Add// Get the list of AIDs in the device

Code example:

```
list.add(EmvAppTag.Application_Identifier_AID_terminal+"0000000000000
0000000000000000000");
list.add(EmvAppTag.Transaction_Currency_Code+"0208");
pos.updateEmvAPP(EMVDataOperation.Add, list);
```

Note: The principle of addition is as follows:

If there is a corresponding AID in the device, the parameter corresponding to the AID is overwritten. ------->> Is there any free space in the device?

(Based on the number of preloaded configurations, if ten groups are preloaded in the device, the addition operation is performed within ten groups)

If there is, then add, no return failure

listener function

```
@Override
public void onReturnGetEMVListResult(String arg0) {
   // TODO Auto-generated method stub
   TRACE.d("onReturnGetEMVListResult(String arg0):" + arg0);

   if (arg0 != null && arg0.length() > 0) {
      statusEditText.setText("The emv list is : " + arg0);
```

```
        }
}
```