Newland AIDC
SCANNING MADE SIMPLE

RFID MODULE

For generic Android device

Version: V2.8

Configuration Handbook

Disclaimer

© 2023 Newland Europe BV. All rights reserved.

Please read the manual carefully before using the product and operate it according to the manual. It is advised that you keep this manual for future reference.

Do not disassemble the device or remove the seal label from the device; doing so will void the product warranty provided by Newland Europe BV.

All pictures in this manual are for reference only, and the actual product may differ.

Regarding product modification and update, Newland Europe BV reserves the right to make changes to any software or hardware to improve reliability, function, or design at any time without notice. The information contained herein is subject to change without prior notice.

The products depicted in this manual may include software copyrighted by Newland Europe BV or a third party. The user, corporation or individual shall not duplicate, in whole or in part, distribute, modify, decompile, disassemble, decode, reverse engineer, rent, transfer, or sublicense such software without prior written consent from the copyright holders.

This manual is copyrighted. No part of this publication may be reproduced, distributed, or used in any form without Newland Europe BV's written permission.

Risk Warning Regarding Unauthorized System Updates:
You should use the Newland-provided tool to update this product's system. Modifying system files by installing a third-party ROM system or using any cracking method may result in product malfunction or data loss and void your warranty.

Newland Europe BV reserves the right to make a final interpretation of the statement above.

Newland Europe BV

Rolweg 25, 4104 AV, Culemborg,
The Netherlands
www.newland-id.com

Newland Europe BV is a subsidiary of Newland Digital Technology Co., Ltd. Our general conditions of Purchase, Sale and Delivery are filed with the Record Office of the Chamber of Commerce of Utrecht, The Netherlands.

K.v.K. H.R. Utrecht / Chamber of Commerce Utrecht: Reg. nr. 17109876

Contents

Contents 0
About This Manual 1
Set Interface 1
Parameters 2
Antenna Power 2
GEN2X 4
Region & Frequency 5
Session 7
Q Value 7
Write Mode【Only applicable to URM_R2/URM_E7】 8
Max Length【Only applicable to URM_R2/URM_E7】 8
Target 9
Gen2 Tag Encoding 9
1.【Only applicable to URM_R2】: 9
2.【Only applicable to URM_E7】: 10
3.【Only applicable to URM300/URF100】: 11
4.【Only applicable to URM500/URF520/URF120】: 11
5.【Only applicable to N6110】: 12
Filter 13
Additional Data 15
Inventory Portfolio Policy 16
Supported models: 【URM_R2】 16
Supported models: 【URM_E7/URM500/URM300/URF520/URF120/N6110】 17
Supported models: 【URF100】 17
Output RSSI 18
Output FREQUENCE 18
FastID 19
Link Settings 19
Inventory 20
Inventory protocol 20
Antenna Group 21
Inventory Timeout and Interval 21
Other Parameters 22
Unique Additional Data 22
Temperature 23
Obtain Module Info 23
Firmware Update 23
GPIO Control【URF520】 24
Antenna Power under Low Battery 26
Data inventory cache output 27
Clear SFExpress tag cache【Support on Newland PDA devices】 28
Filter Duplicate Tags 【Support on Newland PDA Devices】 28
Auto connect on boot【Support on Newland PDA Devices】 28
RSSI Filter 29

About This Manual

This manual is developer guide to call RFID parameter setting interfaces for models URM_R2、URM_E7、URM300、URM500、URF520、URF120 and URF100. Developers may refer to this handbook for configuration details.

Set Interface

Set RFID module parameters like inventory parameters, antenna power parameters, additional data parameters and others. Reading and writing tag performance vary with different parameters.

Interface definition

Class: com.nlscan.uhf.lib.UHFManager	
Return Value	Method
static UHFManager	***getInstance(Context context)*** To get instance
UHFReader.READER_STATE	***setParam(String paramKey,*** ***String paramName,*** ***String sValue)*** Parameters setting Parameter: paramKey function identifier (Please refer to corresponding model for parameter identifier.) Parameter: paramName parameter name (Please refer to corresponding model for parameter identifier.) Parameter: sValue, Return value: (@See ***UHFReader.READER_STATE***)
String	***getParam(String paramKey,*** ***String paramName,*** ***String sDefaultValue)*** Parameter: paramKey, function identifier (Please refer to corresponding model for parameter identifier.) Parameter: paramName, parameter name (Please refer to corresponding model for parameter identifier.) Parameter: ***sDefaultValue*** Return value:String

Example:

```
//get UHFManager instance
UHFManager mUHFMgr = UHFManager.getInstance(mContext);

// Antenna parameter format
/*[{
        "antid": 1,
        "readPower": 33,
        "writePower": 33,
        "enable": true
}, {
        "antid": 2,
        "readPower": 33,
        "writePower": 33,
        "enable": true
}, {
        "antid": 3,
        "readPower": 33,
        "writePower": 33,
        "enable": false
```

```
}, {
        "antid": 4,
        "readPower": 33,
        "writePower": 33,
        "enable": false
}]*/


//----set antenna power
JSONArray jsAntArray = new JSONArray();
try {
        JSONObject jobj = new JSONObject();
        jobj.put("antid", 1);//Ant ID
        jobj.put("readPower", 2700); // read power
        jobj.put("writePower", 2000); // write power
        jobj.put("enable", true); ///true - enable the antenna, false - disable the antenna
        jsAntArray.put(jobj);
} catch (Exception e) {
}
String sAntPowerValue = jsAntArray.toString();
String paramKey = "RF_ANTPOWER";
String paramName = "PARAM_RF_ANTPOWER";
UHFReader.READER_STATE er = mUHFMgr.setParam(paramKey, paramName, sAntPowerValue);
if( er == UHFReader.READER_STATE.OK_ERR)
        //success.....
else
        //fail.....


//----get antenna power
//load “reader transmit power JSONArray ”
String paramKey = "RF_ANTPOWER";
String paramName = "PARAM_RF_ANTPOWER";
String sJsonValue = mUHFMgr.getParam(paramKey, paramName, "");
JSONArray jsArray = new JSONArray(sValue);
int len = jsArray.length();
for(int i =0 ;i < len;i++)
{
        JSONObject jobj = jsArray.optJSONObject(i);
        int antid = jobj.optInt("antid");//Ant ID
        short readPower = (short)jobj.optInt("readPower");// read power
        short writePower = (short)jobj.optInt("writePower");// write power
        boolean antEnable = jobj.optBoolean("enable"); // true - enable the antenna, false - disable the antenna
}
```

Parameters

Antenna Power

Enable/Disable Antenna

Set Antenna Power Parameters:

- URM_R2(Power: 5 ~ 30, Number of antennas: 1)
- URF100(Power: 1 ~ 30, Number of antennas: 1）
- URM300(Power: 1 ~ 30, Number of antennas: 1）
- URM _E7(Power: 5 ~ 33, Number of antennas: 1)
- URM500(Power: 1 ~ 33, Number of antennas: 1)
- URF520(Power : 1 ~ 33, Number of antennas: 4）
- URF120(Power : 1 ~ 33, Number of antennas: 4）
- N6110(Power : 5 ~ 33, Number of antennas:1）

<table>
<tr><th>Power</th><th>paramKey</th><th>paramName</th><th>Value Type</th><th>Value</th></tr>
<tr><td rowspan="2">Power</td><td>RF_ANTPOWER</td><td>PARAM_RF_ANTPOWER</td><td>STRING</td><td>Set multiple antennas’ power and you will get multiple JSON objects from string in JSONArray format.
e.g.:
[
{
"antid":1,
"readPower":26,
"writePower":27,
"enable":true
},
...
]

Antid: Antenna ID, default 1
readPower: Read power（INT）
writePower: Write power（INT）
enable: Enable/disable antenna
● Single antenna: cannot be disabled.
● For multi-antenna setups, at least one antenna must remain active; disabling all antennas is not allowed.</td></tr>
<tr><td colspan="4">Set parameters:

// JSON format string for antenna parameters

/*[{
"antid": 1,
"readPower": 33,
"writePower": 33,
"enable": true
}, {
"antid": 2,
"readPower": 33,
"writePower": 33,
"enable": true
}, {
"antid": 3,
"readPower": 33,
"writePower": 33,
"enable": false
}, {
"antid": 4,
"readPower": 33,
"writePower": 33,
"enable": false
}]*/

UHFManager mUHFMgr = UHFManager.getInstance(mContext);
JSONArray jsAntArray = new JSONArray();
try {
JSONObject jobj_1 = new JSONObject();
jobj_1.put("antid", 1);//Ant ID
jobj_1.put("readPower", 30);//read power
jobj_1.put("writePower", 20);//write power
jobj_1.put("enable", true);// true – Enable the antenna

JSONObject jobj_2 = new JSONObject();
jobj_2.put("antid", 2);//Ant ID
jobj_2.put("readPower", 30);//read power
jobj_2.put("writePower", 20);//write power
jobj_2.put("enable", false);// false – Disable Antenna</td></tr>
</table>

```
        //.....

                jsAntArray.put(jobj_1);
                jsAntArray.put(jobj_2);
        //.....
} catch (Exception e) {
}
String sAntPowerValue = jsAntArray.toString();
String paramKey = "RF_ANTPOWER";
String paramName = "PARAM_RF_ANTPOWER";
UHFReader.READER_STATE er = mUHFMgr.setParam(paramKey,
  paramName, sAntPowerValue);
if( er == UHFReader.READER_STATE.OK_ERR)
                //success.....
else
                //fail.....
```

Get current value:

```
UHFManager mUHFMgr = UHFManager.getInstance(mContext);
String paramKey = "RF_ANTPOWER";
String paramName = "PARAM_RF_ANTPOWER";
String sJsonValue = mUHFMgr.getParam(paramKey, paramName, “”);
JSONArray jsArray = new JSONArray(sValue);
int len = jsArray.length();
for(int i =0 ;i < len;i++)
{
                JSONObject jobj = jsArray.optJSONObject(i);
                int antid = jobj.optInt("antid");//Ant ID
                short readPower = (short)jobj.optInt("readPower");//read power
                short writePower = (short)jobj.optInt("writePower");//write power
                boolean antEnable = jobj.optBoolean("enable"); // true –enable antenna, false – disable antenna
}
```

GEN2X

【Not applicable to URM_E7】

【Not applicable to URM_R2】

Read and interpret tags adhering to different GEN protocols.

Name	paramKey	paramName	Value Type	Value
GEN2X	***GEN2X***	***PARAM_GEN2X_PROTOCOL***	STRING	0: GEN2 only 1: GEN2 + GEN2X 2: GEN2X only 3: GEN2X + TSN Default: by module.

Set parameters:

```
UHFManager mUHFMgr = UHFManager.getInstance(mContext);
//GEN2+GEN2X：
int gen_value = 1 ;
UHFReader.READER_STATE er = mUHFMgr.setParam("GEN2X",
                        "PARAM_GEN2X_PROTOCOL",
                        String.valueOf(gen_value));

if( er == UHFReader.READER_STATE.OK_ERR)
                //success.....
```

	else *//fail....*
	Get current value: UHFManager mUHFMgr = UHFManager.getInstance(mContext); *//GEN2X* String paramKey = "GEN2X"; String paramName = "PARAM_GEN2X_PROTOCOL"; String sGen = mUHFMgr.getParam(paramKey, paramName, "0");

Region & Frequency

Region

Name	paramKey	paramName	Value Type	Value
Region	***FREQUENCY_REGION***	***PARAM_FREQUENCY_REGION***	STRING	China: 6 North America: 1 Europe: 8 Japan：5 Malaysia：15 All: 255 （***N6110 unsupported***） Default: by module.
Frequency	***FREQUENCY_HOPTABLE***	***PARAM_HTB***	STRING	Obtain available range from region you set. e.g.: North America: 902750 903250 903750 ... It is available to select several values with comma “,” separated. e.g.: “902750,903250,903750, ...”
	Set parameters: UHFManager mUHFMgr = UHFManager.getInstance(mContext); *// region：* ***int*** RG_NA = 1 ; UHFReader.READER_STATE er = mUHFMgr.setParam("FREQUENCY_REGION", "PARAM_FREQUENCY_REGION", String.valueOf(UHFReader.Region_Conf.RG_NA)); *// frequency：* String shtb = "902750,903250,903750"; er = mUHFMgr.setParam("FREQUENCY_HOPTABLE", "PARAM_HTB", shtb); ***if***(er == UHFReader.READER_STATE.OK_ERR) *//success.....* ***else*** *//fail....*			

	Get current value:
	
	UHFManager mUHFMgr = UHFManager.getInstance(mContext);
	// region
	String paramKey = "FREQUENCY_REGION";
	String paramName = "PARAM_FREQUENCY_REGION";
	String sRegion = mUHFMgr.getParam(paramKey,
	paramName, String.valueOf(UHFReader.Region_Conf.RG_NA));
	// fequency
	String paramKey = "FREQUENCY_HOPTABLE";
	String paramName = "PARAM_HTB";
	String sHtb = mUHFMgr.getParam(paramKey, paramName, "");

Session

<table>
<tr><th>Name</th><th>paramKey</th><th>paramName</th><th>Value Type</th><th>Value</th></tr>
<tr><td rowspan="2">Session</td><td>POTL_GEN2_SESSION</td><td>PARAM_POTL_GEN2_SESSION</td><td>STRING</td><td>0: S0
1: S1
2: S2
3: S3

Default: 0（S0）</td></tr>
<tr><td colspan="4">Set parameters:

UHFManager mUHFMgr = UHFManager.getInstance(mContext);
UHFReader.READER_STATE er = mUHFMgr.setParam("POTL_GEN2_SESSION",
"PARAM_POTL_GEN2_SESSION",
"0");
if(er == UHFReader.READER_STATE.OK_ERR)
//success.....
else
//fail.....</td></tr>
<tr><td></td><td colspan="4">Get current value:

UHFManager mUHFMgr = UHFManager.getInstance(mContext);
String paramKey = "POTL_GEN2_SESSION";
String paramName = "PARAM_POTL_GEN2_SESSION";
String sSession = mUHFMgr.getParam(paramKey, paramName, "");
int ses =Integer.parseInt(sSession);</td></tr>
</table>

Q Value

Name	paramKey	paramName	Value Type	Value
Q value	***POTL_GEN2_Q***	***PARAM_POTL_GEN2_Q***	STRING	Value: = -1 Auto = 0 = 1 = 2 = 3 = 4 （***N6110 unsupported***） = 5 = 6 = 7 = 8 = 9 = 10 = 11 = 12 = 13 = 14 = 15 Default: -1（Auto）

	Set parameters: UHFManager mUHFMgr = UHFManager.getInstance(mContext); UHFReader.READER_STATE er = mUHFMgr.setParam("POTL_GEN2_Q", "PARAM_POTL_GEN2_Q", "-1"); ***if***(er == UHFReader.READER_STATE.OK_ERR) *//success.....* ***else*** *//fail.....*
	Get current value: UHFManager mUHFMgr = UHFManager.getInstance(mContext); String paramKey = "POTL_GEN2_Q"; String paramName = "PARAM_POTL_GEN2_Q"; String sQvalue = mUHFMgr.getParam(paramKey, paramName, ”-1”); ***int*** gen2Q = Integer.parseInt(sQvalue);

Write Mode【Only applicable to URM_R2/URM_E7】

Supported models: URM_R2/URM_E7

Name	paramKey	paramName	Value Type	Value
Write mode	***POTL_GEN2_WRITEMODE***	***PARAM_POTL_GEN2_WRITEMODE***	STRING	0: word write 1: block write Default: 0 (word write)
	Set parameters: UHFManager mUHFMgr = UHFManager.getInstance(mContext); UHFReader.READER_STATE er = mUHFMgr.setParam("POTL_GEN2_WRITEMODE", "PARAM_POTL_GEN2_WRITEMODE", "0"); ***if***(er == UHFReader.READER_STATE.OK_ERR) *//success.....* ***else*** *//fail.....*			
	Get current value: UHFManager mUHFMgr = UHFManager.getInstance(mContext); String paramKey = "POTL_GEN2_WRITEMODE"; String paramName = "PARAM_POTL_GEN2_WRITEMODE"; String sValue = mUHFMgr.getParam(paramKey, paramName, "-1"); ***int*** writeMode = Integer.parseInt(sValue);			

Max Length【Only applicable to URM_R2/URM_E7】

Supported models: URM_R2/URM_E7

Name	paramKey	paramName	Value Type	Value
Max Length	***POTL_GEN2_MAXEPCLEN***	***PARAM_POTL_GEN2_MAXEPCLEN***	STRING	(bit address) 96 496 Default: 496

	Set parameters: UHFManager mUHFMgr = UHFManager.getInstance(mContext); UHFReader.READER_STATE er = mUHFMgr.setParam("POTL_GEN2_MAXEPCLEN", "PARAM_POTL_GEN2_MAXEPCLEN", "96"); ***if***(er == UHFReader.READER_STATE.OK_ERR) *//success.....* ***else*** *//fail.....*
	Get current value: UHFManager mUHFMgr = UHFManager.getInstance(mContext); String paramKey = "POTL_GEN2_MAXEPCLEN"; String paramName = "PARAM_POTL_GEN2_MAXEPCLEN"; String sValue = mUHFMgr.getParam(paramKey, paramName, "-1"); ***int*** maxLen = Integer.parseInt(sValue);

Target

Name	paramKey	paramName	Value Type	Value
Write mode	***POTL_GEN2_TARGET***	***PARAM_POTL_GEN2_TARGET***	STRING	0: A 1: B 2: A-B 3: B-A （***N6110 unsupported***） Default: 0 (A)
	Set parameters: UHFManager mUHFMgr = UHFManager.getInstance(mContext); UHFReader.READER_STATE er = mUHFMgr.setParam("POTL_GEN2_TARGET", "PARAM_POTL_GEN2_TARGET", "2"); *//A-B* ***if***(er == UHFReader.READER_STATE.OK_ERR) *//success.....* ***else*** *//fail.....*			
	Get current value: UHFManager mUHFMgr = UHFManager.getInstance(mContext); String paramKey = "POTL_GEN2_TARGET"; String paramName = "PARAM_POTL_GEN2_TARGET"; String sValue = mUHFMgr.getParam(paramKey, paramName, "0"); ***int*** target = Integer.parseInt(sValue);			

Gen2 Tag Encoding

1.【Only applicable to URM_R2】:

Name	paramKey	paramName	Value Type	Value
RF mode	***POTL_GEN2_TAGENCODING***	***PARAM_POTL_GEN2_TAGENCODING***	STRING	0: FM0 1: M2 2: M4

				3: M8 Default: 2 (M4)

	Set parameters:
	
	UHFManager mUHFMgr = UHFManager.getInstance(mContext);
	UHFReader.READER_STATE er = mUHFMgr.setParam("POTL_GEN2_TAGENCODING",
	"PARAM_POTL_GEN2_TAGENCODING",
	"1");// M2
	if(er == UHFReader.READER_STATE.OK_ERR)
	//success.....
	else
	//fail.....			

	Get current value: UHFManager mUHFMgr = UHFManager.getInstance(mContext); String paramKey = "POTL_GEN2_TAGENCODING"; String paramName = "PARAM_POTL_GEN2_TAGENCODING"; String sValue = mUHFMgr.getParam(paramKey, paramName, "2"); ***int*** targetEncode = Integer.parseInt(sValue);			

2.【Only applicable to URM_E7】:

Name	paramKey	paramName	Value Type	Value
RF mode	***POTL_GEN2_TAGENCODING***	***PARAM_POTL_GEN2_TAGENCODING***	STRING	0：FM0 1：M2 2：M4 3：M8 16：PROF0 17：PROF1 18：PROF2 19：PROF3 20：PROF4 21：PROF5 101：RFM_1 103：RFM_3 105：RFM_5 107：RFM_7 111：RFM_11 112：RFM_12 113：RFM_13 115：RFM_15 Default: 107（RFM_7）

	Set parameters:
	
	UHFManager mUHFMgr = UHFManager.getInstance(mContext);
	UHFReader.READER_STATE er = mUHFMgr.setParam("POTL_GEN2_TAGENCODING",
	"PARAM_POTL_GEN2_TAGENCODING",
	"113");// RFM_13
	if(er == UHFReader.READER_STATE.OK_ERR)
	//success.....
	else
	//fail.....

	Get current value: UHFManager mUHFMgr = UHFManager.getInstance(mContext); String paramKey = "POTL_GEN2_TAGENCODING"; String paramName = "PARAM_POTL_GEN2_TAGENCODING"; String sValue = mUHFMgr.getParam(paramKey, paramName, "107"); ***int*** targetEncode = Integer.parseInt(sValue);

3.【Only applicable to URM300/URF100】:

Name	paramKey	paramName	Value Type	Value
RF mode	***POTL_GEN2_TAGENCODING***	***PARAM_POTL_GEN2_TAGENCODING***	STRING	MODE_223 = 7 MODE_222 = 8 MODE_241 = 9 MODE_244 = 10 MODE_285 = 11 Default: 10 （MODE_244）

	Set parameters:
	
	UHFManager mUHFMgr = UHFManager.getInstance(mContext);
	UHFReader.READER_STATE er = mUHFMgr.setParam("POTL_GEN2_TAGENCODING",
	"PARAM_POTL_GEN2_TAGENCODING",
	"11");// MODE_285
	if(er == UHFReader.READER_STATE.OK_ERR)
	//success.....
	else
	//fail.....			

	Get current value: UHFManager mUHFMgr = UHFManager.getInstance(mContext); String paramKey = "POTL_GEN2_TAGENCODING"; String paramName = "PARAM_POTL_GEN2_TAGENCODING"; String sValue = mUHFMgr.getParam(paramKey, paramName, "10"); ***int*** targetEncode = Integer.parseInt(sValue);			

4.【Only applicable to URM500/URF520/URF120】:

Name	paramKey	paramName	Value Type	Value
RF mode	***POTL_GEN2_TAGENCODING***	***PARAM_POTL_GEN2_TAGENCODING***	Set value: STRING Get value: INT[] The 1st value	MODE_103 = 0 MODE_302 = 1 MODE_120 = 2 MODE_323 = 3 MODE_202 = 4 MODE_345 = 5 MODE_344 = 6 MODE_223 = 7

				MODE_222 = 8 MODE_241 = 9 MODE_244 = 10 MODE_285 = 11 Default: 10 (MODE_244)

	Set parameters:
	
	UHFManager mUHFMgr = UHFManager.getInstance(mContext);
	UHFReader.READER_STATE er = mUHFMgr.setParam("POTL_GEN2_TAGENCODING",
	"PARAM_POTL_GEN2_TAGENCODING",
	"11");// *MODE_285*
	if(er == UHFReader.READER_STATE.OK_ERR)
	//success.....
	else
	//fail.....			

	Get current value: UHFManager mUHFMgr = UHFManager.getInstance(mContext); String paramKey = "POTL_GEN2_TAGENCODING"; String paramName = "PARAM_POTL_GEN2_TAGENCODING"; String sValue = mUHFMgr.getParam(paramKey, paramName, "10"); ***int*** targetEncode = Integer.parseInt(sValue);			

5.【Only applicable to N6110】:

Name	paramKey	paramName	Value Type	Value
RF mode	***POTL_GEN2_TAGENCODING***	***PARAM_POTL_GEN2_TAGENCODING***	STRING	0 = Tari=25us,FM0,LHF=40KHz 1 = Tari=25us,Miller4,LHF=250KHz 2 = Tari=25us,Miller4,LHF=300KHz 3 = Tari=6.25us,FM0,LHF=400KHz 4 = Tari=25us,Miller4,LHF=320KHz 5 = Tari=6.25us，FM0，LHF=640KHz 6 = Tari=20us，Miller2，LHF=320KHz 7 = Tari=6.25us，Miller4，LHF=400KHz 8 = Tari=12.5us,Miller4,LHF=250KHz 9 = Tari=6.25us,Miller4,LHF=250KHz 10 = Tari=12.5us,Miller4,LHF=300KHz 11 = Tari=6.25us,Miller4,LHF=300KHz 12 = Auto Default: 6 (Tari=20us，Miller2，LHF=320KHz)
	Set parameters: UHFManager mUHFMgr = UHFManager.getInstance(mContext); UHFReader.READER_STATE er = mUHFMgr.setParam("POTL_GEN2_TAGENCODING", "PARAM_POTL_GEN2_TAGENCODING", "1");*//Tari=25us,Miller4,LHF=250KHz* ***if***(er == UHFReader.READER_STATE.OK_ERR) *//success.....* ***else*** *//fail.....*			

	Get current value: UHFManager mUHFMgr = UHFManager.getInstance(mContext); String paramKey = "POTL_GEN2_TAGENCODING"; String paramName = "PARAM_POTL_GEN2_TAGENCODING"; String sValue = mUHFMgr.getParam(paramKey, paramName, "10"); ***int*** targetEncode = Integer.parseInt(sValue);

Filter

After configuring the filter function, restrictions will apply to inventory checks, read/write operations, locking, and destruction activities:

- Matching Filter: Operations will only be conducted on tags that meet the specified criteria.
- Non-Matching Filter: Operations may be performed on all tags except those that meet the specified criteria.

Name	paramKey	paramName	Value Type	Value

Filter	***TAG_FILTER***	***PARAM_TAG_FILTER***	STRING	String in JSON format
				e.g.:
				{
				"fdata":"0A0B0D0E0F",
				"flen":"40",
				"bank":"1".
				"startaddr":"32",
				"isInvert":"1"
				}
				
				Field explanation:
				Fdata: Filter data (hexadecimal string）
				Flen: Filter data length (bits）
				Bank: Filter data memory bank
				Bank type class：
				UHFReader.BANK_TYPE.EPC
				UHFReader.BANK_TYPE.TID
				UHFReader.BANK_TYPE.USER
				
				Startaddr: Start address（bit address）
				EPC-bank start address：32 （bit）
				Other-bank start address：0 （bit）
				
				isInvert: Match/unmatch (0-match, 1-unmatch）

Set parameters:

```
UHFManager mUHFMgr = UHFManager.getInstance(mContext);
JSONObject jsItem = new JSONObject();
int bank = UHFReader.BANK_TYPE.EPC; //EPC Bank
String sHexFdata = "0A0B0D0E0F";
int ln=sHexFdata.length();
if(ln == 1 || ln % 2 == 1)
   ln++;
int flen = (ln/2)*8; //bits
int startaddr = 32; //32 address bit
int isInvert = 0; //0-Match
jsItem.put("bank", bank);
jsItem.put("fdata", sHexFdata);
jsItem.put("flen", flen);
jsItem.put("startaddr",startaddr);
jsItem.put("isInvert", isInvert);
String sValue = jsItem.toString();
//Set filter
UHFReader.READER_STATE er = mUHFMgr.setParam("TAG_FILTER",
                        "PARAM_TAG_FILTER",
                        sValue);
//Remove filter
er = mUHFMgr.setParam("TAG_FILTER",
                        "PARAM_TAG_FILTER",
                        null);
if( er == UHFReader.READER_STATE.OK_ERR)
                //success.....
else
                //fail.....
```

Get current value:

```
UHFManager mUHFMgr = UHFManager.getInstance(mContext);
String paramKey = "TAG_FILTER";
String paramName = "PARAM_TAG_FILTER";
String sValue = mUHFMgr.getParam(paramKey, paramName, "");
JSONObject jsItem = new JSONObject(sValue);
int bank = jsItem.optInt("bank");
String sHexFdata = jsItem.optString("fdata");
int flen = jsItem.optInt("flen");
int startaddr = jsItem.optInt("startaddr");
int isInvert = jsItem.optInt("isInvert");
```

Additional Data

The inventory results will output data from the EPC partition along with data from another partition, such as the TID partition.

Note: The output of the additional data may be inconsistent and not always readable; multiple attempts may be required to successfully retrieve it.

<table>
<tr><th>Name</th><th>paramKey</th><th>paramName</th><th>Value Type</th><th>Value</th></tr>
<tr><td rowspan="3">Additional Data</td><td>TAG_EMBEDEDDATA</td><td>PARAM_TAG_EMBEDEDDATA</td><td>STRING</td><td>String in JSON format
e.g.:
{
"startaddr":"2",
"bytecnt":"4",
"bank":"1".
"accesspwd":"00000001"
}

Field explanation:
Startaddr: Start address (unit: block, that is 16-bit address)
bytecnt : Number of bytes
bank: memory bank (e.g. TID bank=2)
accesspwd: Access password (hexadecimal string, you don’t need to set this item if access password is not set.)</td></tr>
<tr><td colspan="4">Set parameters:

UHFManager mUHFMgr = UHFManager.getInstance(mContext);
int bank = UHFReader.BANK_TYPE.TID; //TID Bank
int startaddr = 2;// Startaddr: Start address (unit: block, that is 16-bit address)
int bytecnt = 4; // Number of bytes
String sHexAccesspwd = "00000001";// accesspwd, hexadecimal
JSONObject jsItem = new JSONObject();
jsItem.put("bank", bank);
jsItem.put("startaddr", startaddr);
jsItem.put("bytecnt", bytecnt);
jsItem.put("accesspwd", sHexAccesspwd);
String sValue = jsItem.toString();
//Set additional data
UHFReader.READER_STATE er = mUHFMgr.setParam("TAG_EMBEDEDDATA",
"PARAM_TAG_EMBEDEDDATA",
sValue);
//Remove additional data
er = mUHFMgr.setParam("TAG_EMBEDEDDATA",
"PARAM_TAG_EMBEDEDDATA",
null);
if(er == UHFReader.READER_STATE.OK_ERR)
//success.....
else
//fail.....</td></tr>
<tr><td colspan="4">Get current value:

UHFManager mUHFMgr = UHFManager.getInstance(mContext);
String paramKey = "TAG_EMBEDEDDATA";
String paramName = "PARAM_TAG_EMBEDEDDATA";
String sValue = mUHFMgr.getParam(paramKey, paramName, ””);
JSONObject jsItem = new JSONObject(sValue);
int bank = jsItem.optInt("bank");
String sHexAccesspwd = jsItem.optString("accesspwd");
int bytecnt = jsItem.optInt("bytecnt");
int startaddr = jsItem.optInt("startaddr");</td></tr>
</table>

Inventory Portfolio Policy

Supported models: 【URM_R2】

- Single-label mode
- Multi-label mode

<table>
<tr><th>Name</th><th>paramKey</th><th>paramName</th><th>Value Type</th><th>Value</th></tr>
<tr><td rowspan="3">Inventory Policy</td><td>INV_QUICK_MODE</td><td>PARAM_INV_QUICK_MODE</td><td>STRING</td><td>Turn on：1
Turn off：0

Default: 0（Turn off：Normal mode）</td></tr>
<tr><td>POTL_GEN2_SESSION</td><td>PARAM_POTL_GEN2_SESSION</td><td>STRING</td><td>GEN2 Session
S1：1 (Multi-label mode, suitable for large batches of labels)

S0: 0 (Single label mode, suitable for labels within dozens)

Default: S0</td></tr>
<tr><td colspan="3">Set parameters:

UHFManager mUHFMgr = UHFManager.getInstance(mContext);
// Turn on quick mode
UHFReader.READER_STATE er = mUHFMgr.setParam("INV_QUICK_MODE",
"PARAM_INV_QUICK_MODE",
"1");

//Turn on GEN2 Session，only S1 or S0
//Turn on GEN2 Session: S1
er = mUHFMgr.setParam("POTL_GEN2_SESSION",
"PARAM_POTL_GEN2_SESSION",
"1");
// Turn on GEN2 Session : S0
er = mUHFMgr.setParam("POTL_GEN2_SESSION",
"PARAM_POTL_GEN2_SESSION",
"0");
if(er == UHFReader.READER_STATE.OK_ERR)
//success.....
else
//fail.....</td></tr>
<tr><td colspan="3">Get current value:

UHFManager mUHFMgr = UHFManager.getInstance(mContext);

//Quick mode enable
String paramKey = "INV_QUICK_MODE";
String paramName = "PARAM_INV_QUICK_MODE";
String sValue = mUHFMgr.getParam(paramKey, paramName, "0");
int quickModeEnable = Integer.parseInt(sValue);

//Session0 or Session1
String paramKey = "POTL_GEN2_SESSION";
String paramName = "PARAM_POTL_GEN2_SESSION";
String sValue = mUHFMgr.getParam(paramKey, paramName, "0");
int ses = Integer.parseInt(sValue);</td></tr>
</table>

Supported models: 【URM_E7/URM500/URM300/URF520/URF120/N6110】

- Single-label mode
- Quickly mode
- Multi-label mode

<table>
<tr><th>Name</th><th>paramKey</th><th>paramName</th><th>Value Type</th><th>Value</th></tr>
<tr><td rowspan="3">Inventory Policy</td><td>INV_POLICY</td><td>PARAM_INV_POLICY</td><td>STRING</td><td>Single-label Mode：0
Quickly Mode：1
Multi-label Mode：2

Default: 1(Quickly mode)</td></tr>
<tr><td colspan="4">Set parameters:

// Set Multi-label Mode
UHFReader.READER_STATE er = mUHFMgr.setParam("INV_POLICY",
"PARAM_INV_POLICY",
"2");
if(er == UHFReader.READER_STATE.OK_ERR)
//success.....
else
//fail.....</td></tr>
<tr><td colspan="4">Get current value:

UHFManager mUHFMgr = UHFManager.getInstance(mContext);
String paramKey = "INV_POLICY";
String paramName = "PARAM_INV_POLICY";
String sValue = mUHFMgr.getParam(paramKey, paramName, "1");
int invMode = Integer.parseInt(sValue);</td></tr>
</table>

Supported models: 【URF100】

- Single-label mode

<table>
<tr><th>Name</th><th>paramKey</th><th>paramName</th><th>Value Type</th><th>Value</th></tr>
<tr><td rowspan="3">Inventory Policy</td><td>INV_POLICY</td><td>PARAM_INV_POLICY</td><td>STRING</td><td>Single-label Mode：0

Default: 0 (Single label Mode)</td></tr>
<tr><td colspan="4">Set parameters:

UHFManager mUHFMgr = UHFManager.getInstance(mContext);
// Single-label Mode
UHFReader.READER_STATE er = mUHFMgr.setParam("INV_POLICY",
"PARAM_INV_POLICY",
"0");
if(er == UHFReader.READER_STATE.OK_ERR)
//success.....
else
//fail.....</td></tr>
<tr><td colspan="4">Get current value:

UHFManager mUHFMgr = UHFManager.getInstance(mContext);
String paramKey = "INV_POLICY";
String paramName = "PARAM_INV_POLICY";
String sValue = mUHFMgr.getParam(paramKey, paramName, "0");
int invMode = Integer.parseInt(sValue);</td></tr>
</table>

Output RSSI

（***N6110 unsupported***）

<table>
<tr><th>Name</th><th>paramKey</th><th>paramName</th><th>Value Type</th><th>Value</th></tr>
<tr><td rowspan="3">RSSI</td><td>INV_FIELD_RSSI</td><td>PARAM_INV_FIELD_RSSI</td><td>STRING</td><td>1：Output RSSI
0：Not output RSSI</td></tr>
<tr><td colspan="4">// Output RSSI when check the inventory：
UHFManager mUHFMgr = UHFManager.getInstance(mContext);
UHFReader.READER_STATE er = mUHFMgr.setParam("INV_FIELD_RSSI",
"PARAM_INV_FIELD_RSSI",
"1");// Output RSSI

if(er == UHFReader.READER_STATE.OK_ERR)
//success
else
//fail</td></tr>
<tr><td colspan="4">Get current value:

UHFManager mUHFMgr = UHFManager.getInstance(mContext);
String paramKey = "INV_FIELD_RSSI";
String paramName = "PARAM_INV_FIELD_RSSI";
String sValue = mUHFMgr.getParam(paramKey, paramName, "0");
int invRSSI = Integer.parseInt(sValue);</td></tr>
</table>

Output FREQUENCE

（***N6110 unsupported***）

<table>
<tr><th>Name</th><th>paramKey</th><th>paramName</th><th>Value Type</th><th>Value</th></tr>
<tr><td rowspan="3">FREQUENCE</td><td>INV_FIELD_FREQUENCE</td><td>PARAM_INV_FIELD_FREQUENCE</td><td>STRING</td><td>1：Output FREQUENCE
0：Not output FREQUENCE</td></tr>
<tr><td colspan="4">// Output FREQUENCE when check the inventory：
UHFManager mUHFMgr = UHFManager.getInstance(mContext);
UHFReader.READER_STATE er = mUHFMgr.setParam("INV_FIELD_FREQUENCE",
"PARAM_INV_FIELD_FREQUENCE",
"1");// Output FREQUENCE
if(er == UHFReader.READER_STATE.OK_ERR)
//success
else
//fail</td></tr>
<tr><td colspan="4">Get current value:

UHFManager mUHFMgr = UHFManager.getInstance(mContext);
String paramKey = "INV_FIELD_FREQUENCE";
String paramName = "PARAM_INV_FIELD_FREQUENCE";
String sValue = mUHFMgr.getParam(paramKey, paramName, "0");
int invFrequence = Integer.parseInt(sValue);</td></tr>
</table>

FastID

When the FastID function is turned on, the TID data will be returned together with the EPC data (with support from the tag itself, URM_E7 is supported)

<table>
<tr><th>Name</th><th>paramKey</th><th>paramName</th><th>Value Type</th><th>Value</th></tr>
<tr><td rowspan="2">FASTID</td><td>FAST_ID</td><td>PARAM_FAST_ID</td><td>STRING</td><td>1: Enable
0: Disable

Default：0 (Disable)</td></tr>
<tr><td colspan="4">Set parameters:

UHFManager mUHFMgr = UHFManager.getInstance(mContext);
//Enale fastid
UHFReader.READER_STATE er = mUHFMgr.setParam("FAST_ID",
"PARAM_FAST_ID",
"1");
if(er == UHFReader.READER_STATE.OK_ERR)
//success.....
else
//fail.....</td></tr>
<tr><td></td><td colspan="4">Get current value:

UHFManager mUHFMgr = UHFManager.getInstance(mContext);
String paramKey = "FAST_ID";
String paramName = "PARAM_FAST_ID";
String sValue = mUHFMgr.getParam(paramKey, paramName, "0");
int fastId = Integer.parseInt(sValue);</td></tr>
</table>

Link Settings

Supported models: URM300

<table>
<tr><th>Name</th><th>paramKey</th><th>paramName</th><th>Value Type</th><th>Value</th></tr>
<tr><td>link Setting</td><td>URM_ProtocolCfg_Gen2</td><td>PARAM_URM_ProtocolCfg_Gen2</td><td>STRING</td><td>{
"tari":1,
"blf":19,
"coding":30
“trext”:30
}
Value：
Tari:
6.25us:0
12.5us:1
25us:2

Blf:
40KHz:0
160KHz:6
213KHz:8
256KHz:9
320KHz:12
640KHz:15

Coding :
em> FM0:0
miller2:1
miller4:2</td></tr>
</table>

				miller8:3 Trext: preYes:1 preNO:0

Set parameters:

```
JSONObject jo = new JSONObject();
try {
    jo.put("tari", t);
    jo.put("blf", b);
    jo.put("coding", c);
    jo.put("trext", tr);
    UHFManager mUHFMgr = UHFManager.getInstance(mContext);
    UHFReader.READER_STATE er = mUHFMgr.setParam("URM_ProtocolCfg_Gen2",
                                                 "PARAM_URM_ProtocolCfg_Gen2",
                                                 jo.toString());
    if( er == UHFReader.READER_STATE. OK_ERR)
            //success.....
    else
            //fail.....

} catch (JSONException e) {
    e.printStackTrace();
}
```

Get current value:

```
UHFManager mUHFMgr = UHFManager.getInstance(mContext);
String paramKey = "URM_ProtocolCfg_Gen2";
String paramName = "PARAM_URM_ProtocolCfg_Gen2";
String cfg = mUHFMgr.getParam(paramKey, paramName, "");

try {
    JSONObject jo = new JSONObject(cfg);
    int tari = hexStr2int(jo.optString("tari"));
    int blf = hexStr2int(jo.optString("blf"));
    int coding = hexStr2int( jo.optString("coding"));
    int trext = hexStr2int( jo.optString("trext"));
} catch (JSONException e) {
    e.printStackTrace();
}
```

Inventory

Inventory protocol

(Supported models: URM_R2/URM_E7)

Name	paramKey	paramName	Value Type	Value
Inventory protocol	***TAG_INVPOTL***	***PARAM_TAG_INVPOTL***	STRING	NONE: 0 ISO180006B: 3 GEN2: 5 ISO180006B_UCODE: 6 IPX64: 7 IPX256: 8 It is available to select several values with comma “,” separated.

				Default: 5（GEN2）
	Set parameters: UHFManager mUHFMgr = UHFManager.getInstance(mContext); UHFReader.READER_STATE er = mUHFMgr.setParam("TAG_INVPOTL", "PARAM_TAG_INVPOTL", "3,5"); ***if***(er == UHFReader.READER_STATE. OK_ERR) *//success.....* ***else*** *//fail.....*			
	Get current value: UHFManager mUHFMgr = UHFManager.getInstance(mContext); String paramKey = "TAG_INVPOTL"; String paramName = "PARAM_TAG_INVPOTL"; String sValue = mUHFMgr.getParam(paramKey, paramName, "0"); ***int***[] protocls = (***int***[]) sValue.split(","); 			

Antenna Group

(Supported models: URM_R2/URM_E7)

Name	paramKey	paramName	Value Type	Value
Antenna	***ANTS***	***PARAM_ANTS_GROUP***	STRING	Antenna: 1 It is available to select several values with comma “,” separated. Default: 1
	Set parameters: UHFManager mUHFMgr = UHFManager.getInstance(mContext); UHFReader.READER_STATE er = mUHFMgr.setParam("ANTS", "PARAM_ANTS_GROUP", "1"); ***if***(er == UHFReader.READER_STATE.OK_ERR) *//success* ***else*** *//failed*			
	Get current value: UHFManager mUHFMgr = UHFManager.getInstance(mContext); String paramKey = "ANTS"; String paramName = "PARAM_ANTS_GROUP"; String sValue = mUHFMgr.getParam(paramKey, paramName, "1"); ***int***[] ants = sValue.split(",");			

Inventory Timeout and Interval

(Supported models: URM_R2/URM_E7)

Name	paramKey	paramName	Value Type	Value

<table>
<tr><td>Inventory timeout</td><td>INV_TIME_OUT</td><td>PARAM_INV_TIME_OUT</td><td>STRING</td><td>Timeout (ms)
Default 50 ms</td></tr>
<tr><td>Inventory interval</td><td>INV_INTERVAL</td><td>PARAM_INV_INTERVAL_TIME</td><td>STRING</td><td>Interval (ms)
Default 0 ms</td></tr>
<tr><td rowspan="2"></td><td colspan="4">Set parameters:
UHFManager mUHFMgr = UHFManager.getInstance(mContext);
UHFReader.READER_STATE er = mUHFMgr.setParam("INV_TIME_OUT",
"PARAM_INV_TIME_OUT",
"50");
UHFReader.READER_STATE er = mUHFMgr.setParam("INV_INTERVAL",
"PARAM_INV_INTERVAL_TIME",
"0");
if(er == UHFReader.READER_STATE.OK_ERR)
//success
else
//failed</td></tr>
<tr><td colspan="4">Get current value:

UHFManager mUHFMgr = UHFManager.getInstance(mContext);
//Inventory timeout
String paramKey = "INV_TIME_OUT";
String paramName = "PARAM_INV_TIME_OUT";
String sTimeout = mUHFMgr.getParam(paramKey, paramName, "50");
long timeout = Long.parseLong(sTimeout);

// Inventory interval
String paramKey = "INV_INTERVAL";
String paramName = "PARAM_INV_INTERVAL_TIME";
String sInterval = mUHFMgr.getParam(paramKey, paramName, "0");
long interval = Long.parseLong(sInterval);</td></tr>
</table>

Other Parameters

Unique Additional Data

Supported models: URM_R2/URM_E7

<table>
<tr><th>Name</th><th>paramKey</th><th>paramName</th><th>Value Type</th><th>Value</th></tr>
<tr><td rowspan="2">Unique Additional Data</td><td>TAGDATA_UNIQUEBYEMDDATA</td><td>PARAM_TAGDATA_UNIQUEBYEMDDATA</td><td>STRING</td><td>1: Unique
0：Not unique

Default: 0 (Not unique)</td></tr>
<tr><td colspan="4">Set parameters:

UHFManager mUHFMgr = UHFManager.getInstance(mContext);
UHFReader.READER_STATE er = mUHFMgr.setParam("TAGDATA_UNIQUEBYEMDDATA",
"PARAM_TAGDATA_UNIQUEBYEMDDATA",
"0");
if(er == UHFReader.READER_STATE. OK_ERR)
//success.....
else
//fail.....</td></tr>
</table>

	Get current value: UHFManager mUHFMgr = UHFManager.getInstance(mContext); String paramKey = "TAGDATA_UNIQUEBYEMDDATA"; String paramName = "PARAM_TAGDATA_UNIQUEBYEMDDATA"; String sValue = mUHFMgr.getParam(paramKey, paramName, "0"); ***int*** unique = Integer.parseInt(sValue);

Temperature

Supported models: URM_R2/URM_E7

Name	paramKey	paramName	Value Type	Value
Temperature	***TEMPTURE***	***PARAM_TEMPTURE***	INT	Module temperature (℃)
	Get current value: UHFManager mUHFMgr = UHFManager.getInstance(mContext); String sValue = mUHFMgr.getParam("TEMPTURE", "PARAM_TEMPTURE", ***null***); ***if***(sValue != ***null*** && TextUtils.isDigitsOnly(sValue)) ***int*** tempture = Integer.parseInt(sValue);			

Obtain Module Info

Name	paramKey	paramName	Value Type	Value
Module Information	MODULE_INFO	PARAM_MODULE_INFO	STRING	Module Information
	Get current module information: UHFManager mUHFMgr = UHFManager.getInstance(mContext); String moduleInfo = mUHFMgr.getParam("MODULE_INFO", "PARAM_MODULE_INFO", ***null***);			

Firmware Update

Support Model: URM_E7

Note: This must be called from non-main thread. After completing the upgrade, it will need to be powered on again to take effect.

Name	paramKey	paramName	Value Type	Value
Firmware Update	***FIRMWARE_UPDATE***	***PARAM_FIRMWARE_UPDATE***	STRING	Firmware Update File Save Path

	// Module firmware update（call from non-main thread）:
	UHFManager mUHFMgr = UHFManager.getInstance(mContext);
	String mRomFilePath = "/sdcard/uhf_image.bin";
	UHFReader.READER_STATE state = uhfMgr.setParam("FIRMWARE_UPDATE",
	"PARAM_FIRMWARE_UPDATE", mRomFilePath);
	if(state == UHFReader.READER_STATE.OK_ERR)
	    //update success
	else
	    //fail...			

Update	Update progress is sent by broadcast			

Progress	//Action
	final String ROM_UPDATE_ACTION = "com.nlscan.intent.action.ROM_UPDATE";
	//Progress：0 – 100
	final String EXTRA_PROGRESS = "progress";
	//Status：
	//"processing" ：update in progress
	// "fail" ：update fail
	// "ok" ：update success
	final String EXTRA_STATE = "state";

GPIO Control【URF520】

1. GPIO Read & Write

Operations on supported GPIO ports (supported in ***SDK v8.4***)

Name	paramKey	paramName	Value Type	Value

GPIO	***GIPO***	***PARAM_GIPO_LIST***	STRING	String in JSON format :
				{
				"GPIO_0":1,
				"GPIO_1":0,
				"GPIO_2":1,
				"GPIO_3":0
				}

```
//Control GPIO（0 ~ 3）state：
UHFManager mUHFMgr = UHFManager.getInstance(mContext);
String sGpio = "{\"GPIO_0\":1,\"GPIO_1\":0,\"GPIO_2\":1,\"GPIO_3\":0}";
UHFReader.READER_STATE er = mUHFMgr.setParam("GIPO",
                                             "PARAM_GIPO_LIST",
                                             sGpio);//GPIO
if( er == UHFReader.READER_STATE.OK_ERR )
   //success
else
   //fail
```

```
//Retrieve GPIO state:
UHFManager mUHFMgr = UHFManager.getInstance(mContext);
String sGpio = mUHFMgr.getParam ("GIPO","PARAM_GIPO_LIST","");//GPIO
try{
        JSONObject jobj = new JSONObject(sGpio);
        int gpio_0 = jobj.optInt("GPIO_0",-1);
        int gpio_1 = jobj.optInt("GPIO_1",-1);
        int gpio_2 = jobj.optInt("GPIO_2",-1);
        int gpio_3 = jobj.optInt("GPIO_3",-1);
        if(gpio_0 != -1)
                //...;
        if(gpio_1 != -1)
                //...;
        if(gpio_2 != -1)
                //...;
        if(gpio_3 != -1)
                //...;
}catch (Exception e){
}
```

2. *GPI Events report*

The GPI state change events for GPI_0 to GPI_3 are reported actively. Clients can get events by broadcast receiver (supported in ***SDK v11.5***).

Name	Parameter	Type	Description
Broadcast Action	nlscan.intent.action.UHF_EVENT_REPORT	STRING	The event reporting broadcast action: This broadcast action to receive GPI events.
Extra event-type	event_report_type	INT	Event type: 1: GPIO event
Extra event-data	event_report_data	Bundle	The event's data, which is the detailed incidental data of the event.

Example code:

```
//Register event report receiver. eg: GPIO event
private void registerEventReportReceiver()
{
        try{
                IntentFilter iFilter = new IntentFilter();
                iFilter.addAction(UHFManager.ACTION_UHF_EVENT_REPORT);
                mContext.registerReceiver(mEventReceiver,iFilter);
                Log.d(TAG,"---registerEventReportReceiver, End.---");
        }catch (Exception e){
        }
}

//unRegister event report receiver. eg: GPIO event
private void unRegisterEventReportReceiver(){
        try{
                mContext.unregisterReceiver(mEventReceiver);
                Log.d(TAG,"---unRegisterEventReportReceiver, End.---");
        }catch (Exception e){
        }
}

private final static String GPI_0 = "GPI_0";
private final static String GPI_1 = "GPI_1";
private final static String GPI_2 = "GPI_2";
private final static String GPI_3 = "GPI_3";

private BroadcastReceiver mEventReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {

                String action = intent.getAction();
                if(!UHFManager.ACTION_UHF_EVENT_REPORT.equals(action))
                        return;

                int eventType = intent.getIntExtra(UHFManager.EXTRA_EVENT_REPORT_TYPE,-1);
                Log.d(TAG,"---Event report receive. eventType:="+eventType);
                if(eventType == UHFManager.EVENT_REPORT_TYPE_GPIO){

                        Bundle eventData = intent.getBundleExtra(UHFManager.EXTRA_EVENT_REPORT_DATA);

                        if(eventData.containsKey(GPI_0))
                                cb_gpi_0.setChecked(eventData.getInt(GPI_0,-1) > 0);
```

```
                                    if(eventData.containsKey(GPI_1))
                                            cb_gpi_1.setChecked(eventData.getInt(GPI_1,-1) > 0);
                                    if(eventData.containsKey(GPI_2))
                                            cb_gpi_2.setChecked(eventData.getInt(GPI_2,-1) > 0);
                                    if(eventData.containsKey(GPI_3))
                                            cb_gpi_3.setChecked(eventData.getInt(GPI_3,-1) > 0);
                            }
                    }
};
```

Antenna Power under Low Battery

Support on Newland PDA device.

Name	paramKey	paramName	Value Type	Value
Enable/Disable	***LOWER_POWER***	***PARAM_LOWER_POWER_DM_ENABLE***	STRING	Enable: 1 Disable: 0 Default: 0 (Disable)
Battery Level	***LOWER_POWER***	***PARAM_LOWER_POWER_LEVEL***	STRING	Range: 0，10，15，20，25，30，35，40，45，50，60 Default:20（20%）
Power	***LOWER_POWER***	***PARAM_LOWER_POWER_DBM***	STRING	Range: 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500, 1600, 1700, 1800, 1900, 2000, 2100, 2200, 2300, 2400, 2500, 2600, 2700, 2800, 2900, 3000 Default: 2000

Set parameters:

```
UHFManager mUHFMgr = UHFManager.getInstance(mContext);
UHFReader.READER_STATE er = mUHFMgr.setParam("LOWER_POWER",
                                                "PARAM_LOWER_POWER_DM_ENABLE",
                                                "1");
er = mUHFMgr.setParam("LOWER_POWER",
                        "PARAM_LOWER_POWER_LEVEL",
                        "25");//25%
er = mUHFMgr.setParam("LOWER_POWER",
                        "PARAM_LOWER_POWER_DBM",
                        "2100");//21db

if( er == UHFReader.READER_STATE. OK_ERR)
        //success.....
else
        //fail.....
```

Get current value:

```
UHFManager mUHFMgr = UHFManager.getInstance(mContext);
String paramKey = "LOWER_POWER";
String paramName = "PARAM_LOWER_POWER_DM_ENABLE";
String sValue = mUHFMgr.getParam(paramKey, paramName, "");
int lowPowerEnable = Integer.parseInt(sValue);
if(lowPowerEnable == 1) //Eanbled
  //...
else //Disabled
  //...
```

	//Battery level paramName = "PARAM_LOWER_POWER_LEVEL" sValue = mUHFMgr.getParam(paramKey, paramName, ""); ***int*** lowBatteryLevel = Integer.parseInt(sValue); *//Ant power on low-power.* paramName = "PARAM_LOWER_POWER_DBM"; sValue = mUHFMgr.getParam(paramKey, paramName, ""); ***int*** lowPowerDBM= Integer.parseInt(sValue);

Data inventory cache output

Cache the inventory data before outputting it to avoid frequent updates on the client side, which could cause interface blocking and hinder interactive operations.

- First come, first output
- Time comes first, output first

Name	paramKey	paramName	Value Type	Value
Cache output	***TAG_SENDER***	***PARAM_TAG_SENDER_TIMEOUT***	STRING	Timeout(ms)：100 Default(ms)：100
		PARAM_TAG_SENDER_COUNT	STRING	Tag number Default：30

Set parameters:

```
UHFManager mUHFMgr = UHFManager.getInstance(mContext);
UHFReader.READER_STATE er = UHFReader.READER_STATE.CMD_FAILED_ERR;
String sTime = "30";//ms
String sCount = "100";
if(!TextUtils.isEmpty(sTime) && TextUtils.isDigitsOnly(sTime))
        er = mUHFMgr.setParam("TAG_SENDER","PARAM_TAG_SENDER_TIMEOUT",sTime);
if(er == UHFReader.READER_STATE.OK_ERR){
        if(!TextUtils.isEmpty(sCount) && TextUtils.isDigitsOnly(sCount))
                er = mUHFMgr.setParam("TAG_SENDER","PARAM_TAG_SENDER_COUNT",sCount);
}
if(er == UHFReader.READER_STATE.OK_ERR)
        //Success
else
        //Fail
```

Get current value:

```
UHFManager mUHFMgr = UHFManager.getInstance(mContext);
String sTime = mUHFMgr.getParam("TAG_SENDER",
                                "PARAM_TAG_SENDER_TIMEOUT",
                                "100");
String sCount = mUHFMgr.getParam("TAG_SENDER",
                                "PARAM_TAG_SENDER_COUNT",
                                "30");
int time = Integer.parseInt(sTime);
int count = Integer.parseInt(sCount);
```

Clear SFExpress tag cache【Support on Newland PDA devices】

Name	paramKey	paramName	Value Type	Value
Clear SFExpress tag cache	***INV_CLEAR_SF_CACHE***	***PARAM_INV_CLEAR_SF_CACHE***	STRING	1：clear

```
//Clear SFExpress tag cache：
UHFManager mUHFMgr = UHFManager.getInstance(mContext);
mUHFMgr.setParam("INV_CLEAR_SF_CACHE",
                 "PARAM_INV_CLEAR_SF_CACHE",
                 "1");//clear
```

Filter Duplicate Tags 【Support on Newland PDA Devices】

Do not output duplicate tag data during inventory.

Name	paramKey	paramName	Value Type	Value
Filter Duplicate Tags	***TAG_DUPLICATE_FILTER***	***PARAM_ENABLE_TAG_DUPLICATE_FILTER***	STRING	0：Output duplicat tag. 1：Do not output duplicat tag.

```
// Do not output duplicate tag data during inventory.
UHFManager mUHFMgr = UHFManager.getInstance(mContext);
mUHFMgr.setParam("TAG_DUPLICATE_FILTER",
                 "PARAM_ENABLE_TAG_DUPLICATE_FILTER",
                 "1");//Do not output
```

Get current value:

```
UHFManager mUHFMgr = UHFManager.getInstance(mContext);
String paramKey = "TAG_DUPLICATE_FILTER";
String paramName = "PARAM_ENABLE_TAG_DUPLICATE_FILTER";
String sValue = mUHFMgr.getParam(paramKey, paramName, "0");
```

Auto connect on boot【Support on Newland PDA Devices】

Automatically connect to the RFID module when the device finishes booting up.

Name	paramKey	paramName	Value Type	Value
Auto connect	***CONNECTION***	***PARAM_CONNECT_ON_BOOT***	STRING	0：No 1：Yes

Set parameters:

```
//Auto connect on boot：
UHFManager mUHFMgr = UHFManager.getInstance(mContext);
mUHFMgr.setParam("CONNECTION",
                 "PARAM_CONNECT_ON_BOOT",
                 "1");//Auto connect on reboot complete
```

	Get current value: UHFManager mUHFMgr = UHFManager.getInstance(mContext); String paramKey = "CONNECTION"; String paramName = "PARAM_CONNECT_ON_BOOT"; String sValue = mUHFMgr.getParam(paramKey, paramName, "0");

RSSI Filter

Tags with RSSI values below the minimum MIN RSSI threshold will not be reported. (***Note: RSSI value output during inventory must be enabled.***)

Name	paramKey	paramName	Value Type	Value
RSSI Filter	***RSSI_FILTER***	***PARAM_MIN_RSSI***	STRING	0：no filter Min Rssi:（eg：-45）
	Set parameters: *//MIN Rssi：* UHFManager mUHFMgr = UHFManager.getInstance(mContext); mUHFMgr.setParam("RSSI_FILTER", "PARAM_MIN_RSSI", "-45");*//Min rssi: -45*			
	Get current value: UHFManager mUHFMgr = UHFManager.getInstance(mContext); String paramKey = "RSSI_FILTER"; String paramName = "PARAM_MIN_RSSI"; String sValue = mUHFMgr.getParam(paramKey, paramName, "0");			

SCANNING MADE SIMPLE

Newland AIDC EMEA
+31 (0) 345 87 00 33
info@newland-id.com

Rolweg 25
4104 AV Culemborg
The Netherlands