marblePORT ver.**1.3**

Programming Guide

1st Edition

AIOI•SYSTEMS CO., LTD.

Revision History

Date	Version	Details
Jan. 13, 2015	1st Edition	

Contents

1. Overview

About this Manual

This manual is used as a guide to develop Android applications. It explains how to call **marblePORT** from your application so that you can develop applications for Smart Tag.

What is marblePORT

marblePORT is a middleware to communicate with Smart Tag. It starts up by utilizing "**Intent**" from the application that wants to communicate with Smart Tag. It communicates with Smart Tag and returns the result to the original application. It is, therefore, possible to create an application just by calling "**Intent**".

Main Features

- **marblePORT** itself is not a library. It is one of the Android applications.
- Since there is no need to write communication programs for Smart Tag, it is easy to create applications for Smart Tag.
- Because there is a built-in Android mechanism to call **marblePORT,** it is not necessary to have indepth knowledge of programming.
- It is compatible with NFC-enabled terminal and "Osaifu-Keitai" (mobile wallet). (automatic identification)

Operation Environment

- Android OS 2.3.3 or later versions
- NFC-enabled device or "Osaifu-Keitai" (mobile wallet) terminal

Applicable Smart Tags

- ST1020 (Smart Tag with 2-inch display)
- ST1027 (Smart Tag with 2.7-inch display)
- SC1029L (SmartCard with 2.9-inch display)

History of Version Updates

Version 1.3

- Compatible with the SC1029L (SmartCard with 2.9-inch display).

Version 1.2

- Determine if there are errors when a process is completed.
- Resends the command when there is an error in sending a command.
- Added a function to save the communication log.

Version 1.1

- Compatible with the ST1027 (Smart Tag with 2.7-inch display).

  Refer to section 4.6 Specify Type of Smart Tag

  3.10 Change the Security Codes (Function Number = 8)

  Added an option in the **Display Image** function to disallow display rewrite.
- Added the Option Parameters

  Section 4.5 Operation of Vibration **upon Completion**

  Section 4.8 Explicit Use of Osaifu-Keitai (mobile wallet)

  Section 4.9 Confirming the Status at the End of Processing
- Regarding the “Register the Display Image” function, added an option to specify the image together.
- Added a function so that with 1 start-up, multiple types of processing can be performed.

  Section 3.11 Complex Processing
- After communication is processed, if the battery is low, a message will be displayed.

Compatibility with the Old Version

The application created with the version 1.0 will also work in version 1.1 ~ 1.3.

Note on Usage

One characteristic of **marblePORT** is the use of **Intent** to start up each process. In the case of multiple processing (e.g., the user’s data is read and the content is displayed accordingl to the content), it is necessary to start up multiple times. Degredation of performance and display quality may occur.

If **marblePORT** cannot meet the execution speed and user interface requirements when incorporating into your business system, consider implementing your own communication processing without using **marblePORT**. You can download sample programs from the web site mentioned above.

The **marblePORT** specifications, including the design of the screen, etc., may be revised without prior notice.

2. Basic Use

This section explains how to use **marblePORT** from external applications.

2.1 Call marblePORT

Explicit **Intent** is used to start **marblePORT.** The main steps are as follows**:**

(1) Create an instance of **Intent** class.

(2) Using **Extra,** specify the function number and parameters for the function number.

(3) Do the option parameter setup as necessary.

(4) Start **Activity** by using the **Intent** object that you created.

When the instance of **Intent** class is created, use the following string as the action.

com.aioisystems.marbleport.EXECUTE

You must specify the function number. Depending on the number, the application (below “host application”) that calls **marblePORT** specifies what you want **marblePORT** to do. Also, there are specific parameters depending on the function number that you specify - - some are required and others are optional.

List of Function Numbers

Function Number	Process Details
0	Show status
1	Get status
2	Read data
3	Write data
4	Show image
5	Register current image
6	Display registered image
7	Clear display
8	Set up security codes
21	Show text

An example below shows the call code to display an image on the display. (Java)

```
Intent intent = new Intent("com.aioisystems.marbleport.EXECUTE");
//Required
intent.putExtra("EXTRA_FUNCTION_NO", 4);
intent.putExtra("EXTRA_BITMAP", bitmap);
//Optional
intent.putExtra("EXTRA_DITHER", true);
startActivity(intent);
```

※ In this example, the values of the parameters and function number are directly specified. Normally, the recommendation is to define the constant. In addition, the **MpHelper** class with the defined constant is available on the website.

When **marblePORT** starts up, a screen to scan Smart Tag will be displayed. When Smart Tag is touched, communication processing begins according to the parameters specified by the application.

Screen waiting for scan

2.2 Find Out the Completion Status of marblePORT Processing

In the default setting, when the communication process is completed normally and the user releases Smart Tag from the reader/writer, the **marblePORT** screen will disappear and return to the host application screen (**Activity).**

To determine if **marblePORT** has finished processing with the host application, the **startActivityForResult** method of Activity class (**android.app.Activity**) is used. Also, it can be implemented by overriding the **onActivityResult** method of **Activity** of the host application.

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
```

With the argument "**resultCode**" passed by this **onActivityResult** method, you can find out the completion status.

The **resultCode** returned by **marblePORT** is as follows:

Value of resultCode

RESULT_OK (=-1)	Requested processing is completed normally.
RESULT_CANCEL (=0)	User did not touch Smart Tag and ended with the return button of **marblePORT**.
RESULT_ERROR_CANCEL (=1) (※unique value of **marblePORT**)	Error occurred during Smart Tag communication process. User ended **marblePORT** with the return button.

2.3 Receive Data from marblePORT

One of the **marblePORT** functions is the transfer of data to the host application (e.g., read data). The data is stored in the **Intent** object with the aforementioned **onActivityResult** and can be retrieved using the extra field name defined by **marblePORT**.

Sample Code: Below is an example showing how to check the completion of read data and get the byte array data from Smart Tag.

```
protected void onActivityResult(int requestCode, int resultCode, Intent data)
{
  super.onActivityResult(requestCode, resultCode, data);
  if(resultCode == RESULT_OK){
    if(requestCode == READ_DATA){
      byte[] userData = data.getByteArrayExtra("EXTRA_USER_DATA");
```

※Names you can use for extra fields will be discussed later.

After the process is completed, you will receive parameters that are specific to the function number used (those that you can get only after executing that function) and parameters that are always available after executing any function.

3. Parameter for Each Function

In this section, **marblePORT's** function specifications and parameters will be discussed.

3.1 Display Status (Function Number = 0)

The status of Smart Tag is displayed after scanning on the **marblePORT** screen. The table below shows the display status. This function will also be executed when **marblePORT** starts up independently.

IDm	Smart Tag identification number
Battery Status	Status of Smart Tag's internal battery power Fine / Normal / Low / Empty
Version Number	Smart Tag's firmware version (displays in hexadecimal)

■Parameter at Startup

Extra Field Name	Type	Required	Explanation
EXTRA_FUNCTION_NO	int	*	Specify 0

■Specific Parameter after Completion

Extra Field Name	Type	Explanation
(None)		

3.2 Get Status (Function Number = 1)

marblePORT is called to get the aforementioned status with the host application. You can get the parameters at completion for this function number as well as the parameters when the processing of all function numbers is finished.

■Parameters at Startup

Extra Field Name	Type	Required	Explanation
EXTRA_FUNCTION_NO	int	*	Specify 1

■Parameters at Completion

Extra Field Name	Type	Explanation
EXTRA_IDM	byte[]	Smart Tag ID (**IDm**)
EXTRA_BATTERY	byte	Internal battery status of Smart Tag 0: Fine 1: Normal 2: Low 3: Empty
EXTRA_TAG_STATUS	byte	Status of Smart Tag obtained last 00h：Initialization status（RESET） F0h：Process completion F1h：Waiting for command to be received (Fnum≠Fsum） F2h：Being processed F3h：Received command error —（Address error） F4h：Received command error —（Length error） F6h：Received command error —（Size error） F7h：Received command error —（Undefined Func.No.） F8h：Parameter error F9th: Security code mismatch FDh：Flash Rom error FEh：Electronic paper display error FFh：Received FeliCa data error
EXTRA_FIRMWARE_VERSION	byte	Smart Tag firmware version

3.3 Read User Data (Function Number = 2)

Data is read from Smart Tag's internal memory (**Free Information Area**). The data is read in bytes. **marblePORT** and Smart Tag do not stipulate any data format. You can determine the data format with an application as necessary.

■Parameters at Startup

Extra Field Name	Type	Required	Explanation
EXTRA_FUNCTION_NO	int	*	Specify 2
EXTRA_START_ADDRESS	int	-	Read start address (ST1020: 0~3071, ST1027/SC1029L: 0~16383, Default=0).
EXTRA_READ_SIZE	int	*	Read data in byte (ST1020:1~3072, ST1027/SC1029L: 1 ~ 16384).

※ If the start address and size or the start address + length exceed the range, an error will be displayed at the start of **marblePORT** and the communication will not be processed.

■Specific Parameters at Completion

Extra Field Name	Type	Required
EXTRA_USER_DATA	byte[]	Read data

3.4 Write User Data (Function Number = 3)

Write data in the internal memory of Smart Tag (**Free Information Area**). There is no rule for data format for Smart Tag. Determine the data format for an application as necessary.

■Parameters at Startup

Extra Field Name	Type	Required	Explanation
EXTRA_FUNCTION_NO	int	*	Specify 3
EXTRA_START_ADDRESS	int	-	Write start address (ST1020: 0~3071, ST1027/SC1029L: 0~16383, default =0)
EXTRA_USER_DATA	byte[]	*	Write data. Write all data in the array. Array size is ST1020:1~3072, ST1027/SC1029L: 1~16384).

※ If the start address and size or the start address + length exceed the range, an error will be displayed at the start of **marblePORT** and the communication will not be processed.

■Specific Parameters at Completion

Extra Field Name	Type	Explanation
(None)		

3.5 Image Display (Function Number = 4)

Display any image on the Smart Tag display.

■Parameters at Startup

Extra Field Name	Type	Required	Explanation
EXTRA_FUNCTION_NO	int	*	Specify 4
EXTRA_BITMAP	Bitmap*1	*	Image data to be displayed. A color image will be changed to black and white. If the image data exceed the display size limit specified for each type, an error will happen.
EXTRA_DITHER	boolean	-	**true:** White-Black conversion by dithering (error diffusion method). **false:** White-Black conversion at threshold (default value).
EXTRA_LOCATION_X	int	-	Image is placed on the X coordinate (0 at the left edge). Default value = 0.
EXTRA_LOCATION_Y	int	-	Image is placed on the Y coordinate (0 at the top edge). Default valaue = 0.
EXTRA_DRAW_MODE	int	-	How to draw the background (margins of the specified image). 0: Fill the background with white. 1: Fill the background with black. 2: Maintain the existing status (default value). 3: Image of the specified registered number.
EXTRA_LAYOUT_NUMBER	int	-	Specify registered number when 3 is specified with EXTRA_DRAW_MODE. Default value = 0 (without specifying the registered number).
EXTRA_UPDATE_DISPLAY	boolean		Specify whether to rewrite or not rewrite the display. Use when sending multiple partial images. (for ST1027/SC1029L only) **true**: Rewrite the display (default value) **false**: Do not rewrite the display

*1: Bitmap = android.graphics.Bitmap

■Specific Parameter at Completion

Extra Field Name	Type	Explanation
(None)		

3.6 Register the Display Image (Function Number = 5)

Register the image displayed on the Smart Tag display of Smart Tag. The registered image can be displayed with “**Display the Registered Image**” (Function Number = 6).

■Parameters at Startup

Extra Field Name	Type	Required	Explanation
EXTRA_FUNCTION_NO	int	*	Specify 5.
EXTRA_LAYOUT_NUMBER	int	*	Specify the number to register. （ST1020: 1~12 / ST1027: 1~128 / SC1029L: 1）.
EXTRA_UPDATE_DISPLAY	boolean		Specify whether to rewrite or not rewrite when you register. (for ST1027/SC1029L only) **true**: Rewrite the display (default value) false: Do not rewrite the display
EXTRA_BITMAP	Bitmap		Specify the image to be registered. If omitted, the image displayed on Smart Tag will be registered.

■Specific Parameters at Completion

Extra Field Name	Type	Explanation
(None)		

3.7 Display the Registered Image (Function Number =6)

Display the image registered with “**Register the Display Image**”. If a number specified is not registered, fill in with white.

■Parameters at Startup

Extra Field Name	Type	Required	Explanation
EXTRA_FUNCTION_NO	int	*	Specify 6.
EXTRA_LAYOUT_NUMBER	int	*	Specify the registration number to display. （ST1020:1~12 / ST1027:1~128 / SC1029L:1）

■Specific Parameter at Completion

Extra Field Name	Type	Explanation
(None)		

3.8 Clear Display (Fuction Number =7)

Clears the Smart Tag display (fill with white).

■ Parameters at Startup

Extra Field Name	Type	Required	Explanation
EXTRA_FUNCTION_NO	int	*	Specify 7

■ **Specific Parameter at Completion**

Extra Field Name	Type	Explanation
(None)		

3.9 Display Text (Function Number =21)

It is easy to display text on the Smart Tag display. The text wraps automatically but does not hyphenate English words. You can start a new line by specifying the CR code (**0x0D**) to the text. If the text does not fit in the space, it will be ignored.

■ **Parameters at Startup**

Extra Field Name	Type	Required	Explanation
EXTRA_FUNCTION_NO	int	*	Specify 21.
EXTRA_DISPLAY_TEXT	String	*	Text to be displayed.
EXTRA_TEXT_SIZE	int	-	Size of the character. Specify greater than 1. Default value = 16.

■**Specific Parmeter at Completion**

Extra Field Name	Type	Explanation
(None)		

3.10 Change the Security Codes (Function Number = 8)

The security codes of the 2.7-inch Smart Tag can be changed. Set up threee 3-byte security codes, one for displaying, one for memory writing, and one for reading from memory in Smart Tag. When security codes are set up, it is necessary to specify the code each time the function is called. If you specify a security code for the calling function that does not match the security code that was set up in Smart Tag, the processing will fail to execute (does not show up as an error).

To check whether the security code has been correctly changed, check the status after the process has been completed. (Refer to parameters at completion in Section 3.2 Get Status (Function Number = 1))

It is not necessary to set up all three security codes 1 - 3. You can set up only the codes that you need. The factory default of the security codes are: **0x30**, **0x30**, **0x30**.

■Startup Parameters

Extra Field Name	Type	Required	Explanation
EXTRA_FUNCTION_NO	int	*	Specify 8
EXTRA_SECURITY_CODE1	byte[]	(*)	For display Security code before the change, 3 bytes
EXTRA_NEW_SECURITY_CODE1	byte[]	(*)	For display Security code after the change, 3 bytes
EXTRA_SECURITY_CODE2	byte[]	(*)	For writing to the memory Security code before the change, 3 bytes
EXTRA_NEW_SECURITY_CODE2	byte[]	(*)	For writing to the memory Security code after the change, 3 bytes
EXTRA_SECURITY_CODE3	byte[]	(*)	For reading the memory Security code before the change, 3 bytes
EXTRA_NEW_SECURITY_CODE3	byte[]	(*)	For reading the memory Security code after the change, 3 bytes

■Parameters at Completion

Extra Field Name	Type	Explanation
(None)		

3.11 Complex Processing

From just one startup, **marblePORT** can perform multiple processes, such as read/write user data, display images, display registered images, and clear display. Possible combinations of

functions are listed below.　Specify the Function Number for the combined functions and required parameters described in this section, and then run **marblePORT**.

■Possible Combinations

To Combine (Execute in this order)	Function Number
Read user data → Display image	31
Read user data → Display registered image	32
Read user data → Clear display	33
Read user data → Write user data	34
Read user data → Write user data → Display image	41
Read user data → Write user data → Display registered image	42
Read user data → Write user data → Clear display	43
Write user data → Display image	51
Write user data → Display registered image	52
Write user data → Clear display	53

※Communication Errors during Process

If a communication error occurs while processing, you can touch Smart Tag again to start reprocessing from the beginning.　However, if read user data is included in the combination, and error occurs after read user data has completed normally, the data will be cashed.　When the reprocess operation is performed, it will skip read data and proceed to execute the next process.

※About the Start Address of Read and Write User Data

When read and write are combined, use the extra field name "**EXTRA_START_ADDRESS**" for the start address for reading, and "**EXTRA_START_ADDRESS2**" for the start address for writing.

4. Option Parameter

This section explains the parameters that you can set up as an option.

4.1 Completion Operation

Specify the operation after Smart Tag is touched and the communication process is completed.

■Specify at Startup

Extra Field Name	Type	Explanation
EXTRA_AUTO_CLOSE	int	0: The screen remains unchanged after completion of process. (To return, press the “Return” button.) 1: Closes when Smart Tag is released from the terminal. (Default value). 2: Closes immediately when processing is finished.

※The same process can be repeated when “0” is specified. Smart Tag can be touched while screen is being displayed.

4.2 Display the Progress Bar

Specify whether or not to display the **Progress Bar** during Smart Tag communication.

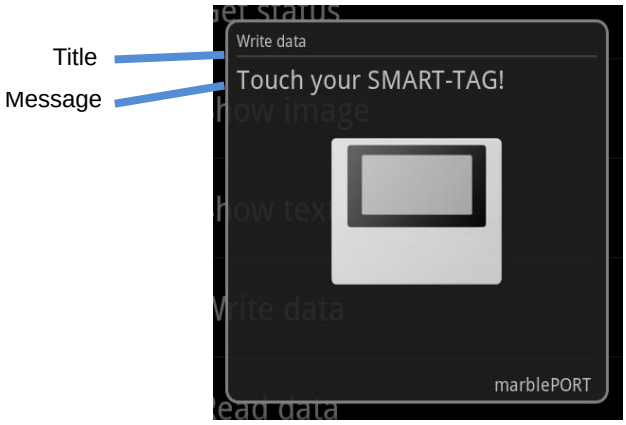
■Specify at Startup

Extra Field Name	Type	Explanation
EXTRA_SHOW_PROGRESS	boolean	true: Display false: Not display (default value)

4.3 Customize the Title / Message

Specify the title and various messages of the **marblePORT** screen. Try to make it a 1-line title. If it does not fit within 1 line, excess characters will be omitted.

The message will wrap at the end. Keep the message to 2 lines. If the message is 3 lines, the message will overlap with the picture of Smart Tag and will be difficult to see.



■Specify at Startup

Extra Field Name	Type	Explanation
EXTRA_TITLE	String	Specify the text of the title column.
EXTRA_MSG_SCAN	String	Specify the display message when waiting for Smart Tag to scan.
EXTRA_MSG_PROCESSING	String	Specify the display message when communication is in progress.
EXTRA_MSG_END	String	Specify the display message when communication is completed.
EXTRA_MSG_IO_ERROR	String	Specify the message when there is a communication error. (The error detail will display at the bottom of the screen.)

4.4 Specifying Instance for Tag Communication (NFC-enabled terminals only)

You can specify **Tag** objects (**android.nfc.Tag**) to communicate with Smart Tag. This applies only when the NFC-enabled terminals are used.

You can specify all 3 fields shown below. Details on how to use these fields will be provided later.

■Specification at Startup

Extra Field Name	Type	Explanation
EXTRA_RELAY_TAG	boolean	Specify **true** when Tag Instance is passed. Otherwise, specify **false** (default value).
EXTRA_ACTIVE_TAG	Tag	**Tag** object to pass. **Tag** must be able to communicate.
EXTRA_IDM	byte[]	**IDm** of Smart Tag that is communicating.

4.5 Operation of Vibration upon Completion

This function specifies whether or not you want Smart Tag to vibrate immediately after **marblePORT** has finished communicating with Smart Tag.

Specification at Startup

Extra Field Name	Type	Explanation
EXTRA_VIBRATE	int	**0**: Do nothing (default value) **1**: Vibrate
EXTRA_VIBRATE_DURATION	int	Specifies the vibration time in millisecond (0 ~ 10,000).

4.6 Specify Type of Smart Tag

You can specify the type of Smart Tag. If you don't specify, it operates as a 2-inch type. When using the 2.7-inch type, specify the option whenever **marblePORT** is called.

Specification at Startup

Extra Field Name	Type	Explanation
EXTRA_SMART_TAG_TYPE	int	**20**: 2-inch type (default value) **27**: 2.7-inch type **29**: 2.9-inch type

4.7 Specify the Security Codes

For the ST1027 (2.7-inch Smart Tag), 3-byte security codes are required if you want to modify the display and read and write in the memory. You must specify in advance the same security code that you used to set up in Smart Tag. The factory default security code is "000"(0x30, 0x30, 0x30).

If you do not specify, Smart Tag will use "000".

Specification at Startup

Extra Field Name	Type	Explanation
EXTRA_SECURITY_CODE1	byte[]	Security code to update the display (display image / clear display / register layout) (3 bytes)
EXTRA_SECURITY_CODE2	byte[]	Security code to write to memory (3 bytes)
EXTRA_SECURITY_CODE3	byte[]	Security code to read from memory(3 bytes)

4.8 Explicit Use of Osaifu-Keitai (mobile wallet)

If the terminal can use both Osaifu-keitai and NFC, **marblePORT** will use NFC over Osaifu-keitai. You can specify Osaifu-keitai to communicate with Smart Tag.

(With Osaifu-keitai, the maximum number of blocks that can be transferred simultaneously is 8. Set up “Simultaneous transport block number” from the **marblePORT** screen according to your need.)

Specification at Startup

Extra Field Name	**Type**	**Explanation**
`EXTRA_USE_FELICA`	`boolean`	**false**: NFC priority (default value) **true**: Use Osaifu-keitai (mobile FeliCa)

4.9 Confirming the Status at the End of Processing

At default (**false**), the status is not confirmed at the end of processing. When set to “**true**”, the status will be confirmed after Smart Tag processing is finished. When completely finished, **marblePORT** screen will then show “**Done**”.

If you confirm the status after the screen update, communication will end because the drawing on the Smart Tag screen has been completed.

Specification at Startup

Extra **Field Name**	**Type**	**Explanation**
`EXTRA_CHECK_STATUS_AFTER_PROCESS`	`boolean`	**false**: Not check (default value) **true**: Check

※From version 1.2, regardless of the option, the status is obtained only once immediately after communication is completed. Normal completion happens only when the status is **0xF0** (normal completion) or **0xF2** (in process). Otherwise, an error will occur and the screen will ask the user to touch Smart Tag again.

5. Practical Uses of marblePORT

This section describes the practical use of **marblePORT**.

5.1 Process Multiple Consecutive Tasks

With just one startup, **marblePORT** is limited in performing various communication processes with Smart Tag. If you want **marblePORT** to perform complex communication with Smart Tag by scanning just once, you need to have **marblePORT** start and end repeatedly while Smart Tag is touched to perform multiple processing.

Some processing of NFC-enabled phones and Osaifu-Keitai (mobile wallet) are done differently.

5.1.1 For the NFC-enabled Phones

You can process several consecutive tasks.

You have to reuse Tag information. You can obtain a Tag object and **IDm** from **Intent** and use them as the parameters for the entire process.

The host application gets the Tag object and **IDm** from **Intent** when the first processing is completed. From the 2nd call and thereafter, these parameters are specified.

■Received information after completion

Extra field name	Type	Explanation
`EXTRA_ACTIVE_TAG`	`Tag`	Tag object that has been using in the process.
`EXTRA_IDM`	`byte[]`	Smart Tag IDm

■Set up parameters for the next process startup

Extra Field Name	Type	Explanation
`EXTRA_RELAY_TAG`	`boolean`	True
`EXTRA_ACTIVE_TAG`	`Tag`	**Tag** object is received when the process is completed.
`EXTRA_IDM`	`byte[]`	Smart Tag IDm

For all processes, except the last process, the parameter "**EXTRA_AUTO_CLOSE**" (behavior of the process completion) must be set to **2** to immediately close the screen.

5.2 Individually Detect Tags (for NFC only)

Normally, you need to scan Smart Tag after **mablePORT** is launched and the host application has started. However, you can have the host application perform the tag detection and have **marblePORT** run only the communication process.

Android SDK and programming knowledge are must-have skills for developing **merblePORT** applications.

(1) In the host application, set to allow receiving NfcF class (**android.nfc.tech.NfcF**) compatible tag by calling **Intent** by Android OS.

(2) Get the **Tag** object (**Extra_Tag**) and Smart Tag ID (**Extra_ID**) from the **Intent** object passed by **onNewIntent** when Smart Tag is detected.

(3) Run **marblePORT** by setting options for **EXTRA_RELAY_TAG, EXTRA_ACTIVE_TAG**, **EXTRA_IDM** and the function number.

(4) **marblePORT** starts processing with the passed **Tag** object and **IDm**.

5.3 Using the Communication Log

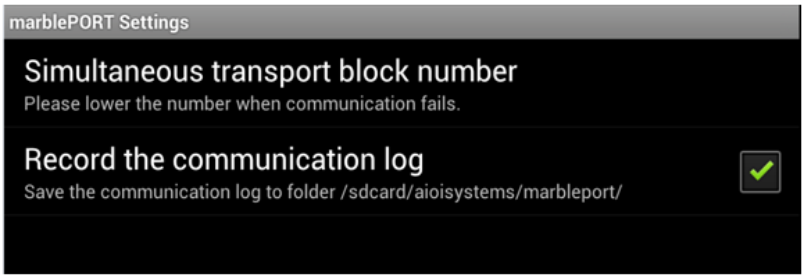
You can save and check part of the content of communication between **marblePORT** and Smart Tag. Only the Smart Tag ID and communication command headers are recorded; images and user data are not recorded. 2MB are required for the log file. If the file is larger than 2MB, 1MB will be deleted from the old log and the new log will be recorded.

Log Activation

The setup screen of **marblePORT** is used to enable and disable the log. Disable is the default setting. To display the setup screen, touch the icon (picture of a wrench) at the bottom of the **marblePORT** screen**.**

Put a check mark for **Record the communication log** to activate. Once the log is enabled, it will remain activated even when **marblePORT** is restarted.

Touch here



Setup screen

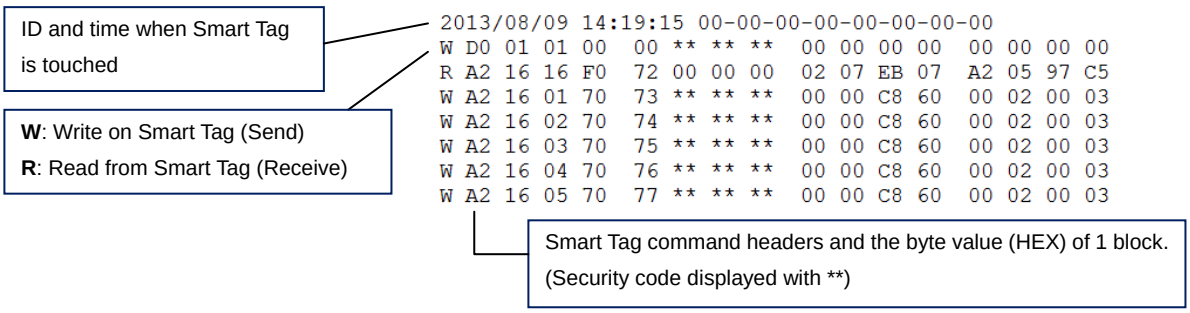
Where to save

The log is created inside "**/sdcard/aioisystems/marbleport/**". You cannot specify the path to save.

Log file name

log.txt	The latest log of approximately 1MB (max.) will be saved. If over 1MB, it will be renamed to "**log.bak.txt**".
log.bak.txt	1MB log of old **log.txt** has been saved.

A sample log file:



Content recorded in the log

- Only Smart Tag commands are recorded. (FeliCa polling command etc. is not recorded.)
- The actual communication is performed with FeliCa commands, but only that portion for Smart Tag commands is recorded.
- If resending occurs at the time a command is sent, a one-line "**retry sending.**" will be recorded just before the resend. If it is determined that there was a sending error, a one-line "**sending error.**" will be recorded.

6. Errors

In this section, what types of errors occur during the processing **marblePORT** will be discussed.

6.1 Error at Startup

For phones that are incompatible with NFC or Osaifu-keitai (NTTDoCoMo mobile phones)

When **marblePORT** is launched, it tries to determine whether the phone is NFC-enabled or Osaifu-ketai enabled starting first with NFC.

First, **marblePORT** runs **getDefaultAdapter** method of **NfcAdapter** class (**android.nfc.NfcAdapter**). If **null** is returned, then the phone is a non NFC-enabled phone.

Next, **marblePORT** tries to start with Osaifu-keitai.

If an error occurs at the start of Osaifu-keitai, SDK (Mobile FeliCa Client) for Osaifu-keitai will display error information.

When both NFC phone and Osaifu-keitai have an error, the **marblePORT** screen will continue to display an error and will not respond even if Smart Tag is touched.

Click the”**Back**” button to return to the host application. You will receive the **resultCode** as **RESULT_ERROR_CANCEL(=1)** through **onActivityResult**.

When NFC is “Disabled” in the device's Settings menu

If the NFC setting is disabled, the “**NFC has been disabled**” message will be displayed when **marblePORT** starts up. (There will be no response when Smart Tag is touched.)

Click Android device’s **“Back**” button to return to the host application. You will receive the **resultCode**, **RESULT_ERROR_CANCEL(=1)** through **onActivityResult**.

After you enable **NFC** in the device's **Settings** screen, you are able to launch **marblePORT**.

6.2 Communication Error

If failure occurs during communication, for example, Smart Tag was released, a communication error message will be displayed and shows the types of errors. An example is shown below.

■Types of Errors

Communication Error	The communication is terminated when the tag has moved out of range. "**Maximum simultaneous transfer block** " means that your "Maximum simultaneous transfer block" setting is too big
Smart tag Error	Abnormal status of Smart Tag. The status variable is displayed. F3-F8, FF: Command Format Error FD, FE, F5: Hardware Failure 70: No error code
NFC Unknown Error	Other unexpected failures of **marblePORT**

※You can capture the error details using LogCat in the Android SDK tool kit.

(**TAG:marblePORT**)

If an error occurs, that error will continue to be displayed no matter what option you choose on the host application. If the error remains on the display, you need to touch Smart Tag again to retry the starting process.

Click the "**back**" button then return to the host application. You will receive the **resultCode** as **RESULT_ERROR_CANCEL(=1)** through **onActivityResult**.

6.3 When Battery is Low

If communication processing is performed when the status of battery capacity indicates low (2 or 3), the screen on **marblePORT** appears as if the communication process completed normally. This is because when the battery capacity is low, communication to process the response is

performed normally, but not the internal processing.
When the battery capacity is 2 or 3 after communication is finished, the following messages will be displayed to show low battery status.

Status of Residual Battery when it is:

- **2**: Battery is getting low
- **3**: Cannot use

After **marblePORT** ends, you can get the status of the battery power capacity with the calling application, **EXTRA_BATTERY**. (Refer to "**2.3 Receive Data from marblePORT**")

※Depending on the operation environment and conditions, even after displaying the above messages, in some cases, it may return to “**0**” or “**1**”.

7. Appendix

7.1 List of Parameters

Specify Function Number

Extra Field Name	Type	Value	Explanation
EXTRA_FUNCTION_NO	int	0: Show status 1: Get status 2: Read data 3: Write data 4: Show image 5: Register current image 6: Show registerd image 7: Clear display 8: Change security code 21: Show text	Function Number

Function Number for Complex Processing

Complex Processing	Function Number
Read user data → Display image	31
Read user data → Display registered image	32
Read user data → Clear display	33
Read user data → Write user data	34
Read user data → Write user data → Display image	41
Read user data → Write user data → Display registered image	42
Read user data → Write user data → Clear display	43
Write user data → Display image	51
Write user data → Display registered image	52
Write user data → Clear display	53

Show Status Parameter

Extra Field Name	Type	Value	Explanation
EXTRA_FUNCTION_NO	int	0	

Get Status

Extra Field Name	Type	Value	Explanation
EXTRA_FUNCTION_NO	int	1	

EXTRA_IDM	byte[]		IDm
EXTRA_BATTERY	byte		Battery Status
EXTRA_TAG_STATUS	byte		Smart Tag Status
EXTRA_FIRMWARE_VERSION	byte		Smart Tag Firmware Version

Read User Data

Extra Field Name	Type	Value	Explanation
EXTRA_FUNCTION_NO	int	2	
EXTRA_START_ADDRESS	int	0~3071 (ST1020) 0~16383 (ST1027/SC1029L) Default value: 0	Start Adress
EXTRA_READ_SIZE	int	1-3072 (ST1020) 1-16384 (ST1027/ SC1029L)	Byte size of read data

EXTRA_USER_DATA	byte[]		Read Data

Write User Data

Extra Field Name	Type	Value	Explanation
EXTRA_FUNCTION_NO	int	3	
EXTRA_START_ADDRESS	int	0~3071 (ST1020) 0~16383 (ST1027/ SC1029L) Default value: 0	Start address
EXTRA_USER_DATA	byte[]		Write data

Display Image

Extra Field Name	Type	Value	Explanation
EXTRA_FUNCTION_NO	int	4	
EXTRA_BITMAP	Bitmap		Bitmap data for display
EXTRA_DITHER	boolean	true: Dithering false: Threshold (default value)	
EXTRA_LOCATION_X	int	0 (default value) – 199 (ST1020) 0-263 (ST1027) 0-299(SC1029L)	X start point Display position (left most is 0)
EXTRA_LOCATION_Y	int	0 (default value) – 95 (ST1020) 0-175 (ST1027) 0-199 (SC1029L)	Y start point Display position (top edge is 0)
EXTRA_DRAW_MODE	int	0: Fill background white 1: Fill background black	Specify background (margins around the specified image)

		2: Keeping current value (defaultl value) 3: specified with image registeration number	
EXTRA_LAYOUT_NUMBER	int	0 (default value): Image registeration number not specified 1 or more: Image registeration number 1-12 (ST1020) 1-128 (ST1027) 1 (SC1029L)	
EXTRA_UPDATE_DISPLAY	boolean	**true**: Rewrite the display (default value) **false**: Do not rewrite the display	For ST1027/SC1029L only

Register Display Image

Extra Field Name	Type	Value	Explanation
EXTRA_FUNCTION_NO	int	5	
EXTRA_LAYOUT_NUMBER	int	1-12 (ST1020) 1-128 (ST1027) 1 (SC1029L)	Registeration number
EXTRA_UPDATE_DISPLAY	boolean	**true**: Rewrite the display (default value) **false**: Do not rewrite the display	For ST1027 only
EXTRA_BITMAP	Bitmap		Specify the image to be registered.

Display Registered Image

Extra Field Name	Type	Value	Explanation
EXTRA_FUNCTION_NO	int	6	
EXTRA_LAYOUT_NUMBER	int	1-12 (ST1020) 1-128 (ST1027) 1 (SC1029L)	Registeration number

Clear Display

Extra Field Name	Type	Value	Explanation
EXTRA_FUNCTION_NO	int	7	

Show Text

Extra Field Name	Type	Value	Explanation
EXTRA_FUNCTION_NO	int	21	
EXTRA_DISPLAY_TEXT	String		Text for display
EXTRA_TEXT_SIZE	int	Default value:16	Size of text

Option - Completion Operation

Extra Field Name	Type	Value	Explanation
EXTRA_AUTO_CLOSE	int	0: Not Close 1:Close after releasing Smart Tag（default value） 2:Close immediately	

Option – Show Progress Bar

Extra Field Name	Type	Value	Explanation
EXTRA_SHOW_PROGRESS	boolean	true: Display false: Not display (default value)	

Option – Customize Title/Message

Extra Field Name	Type	Value	Explanation
EXTRA_TITLE	String		Title
EXTRA_MSG_SCAN	String		Waiting to be scanned
EXTRA_MSG_PROCESSING	String		Communicating
EXTRA_MSG_END	String		Communication completed
EXTRA_MSG_IO_ERROR	String		Communication failure

Option– Specify Tag Instance

Extra Field Name	Type	Value	Explanation
EXTRA_RELAY_TAG	boolean	true: Use false: Not use (default value)	
EXTRA_ACTIVE_TAG	Tag		Tag Object
EXTRA_IDM	byte[]		IDm

Option – Vibrate Upon OperationCompletion

Extra Field Name	Type	Value	Explanation
EXTRA_VIBRATE	int	**0**: Do nothing (default value) **1**: Vibrate	

Option – Specify the Type of Smart Tag

Extra Field Name	Type	Value	Explanation
EXTRA_SMART_TAG_TYPE	int	**20**: 2-inch type (default value) **27**: 2.7-inch type	

Option – Specify Security Code (for ST1027/SC1029L only)

Extra Field Name	Type	Value	Explanation
EXTRA_SECURITY_CODE1	byte[]		Security code for updating the display (3 bytes)

EXTRA_SECURITY_CODE2	byte[]		Security code for writing in memory (3 bytes)
EXTRA_SECURITY_CODE3	byte[]		Security code for reading from memory (3 bytes)

Option – Explicit Use of Osaifu-Keitai (Mobile Wallet)

Extra Field Name	Type	Value	Explanation
EXTRA_USE_FELICA	boolean	**false**: NFC priority (default value) **true**: Use Osaifu-keitai (mobile wallet)	

Option – Confirm the Status at the End of Processing

Extra Field Name	Type	Value	Explanation
EXTRA_CHECK_STATUS_AFTER_PROCESS	boolean	**false**: Do not confirm (default value) **true**: Confirm	

7.2 Version Code of marblePORT

The **marblePORT** version name and code are the following:

Version Name	Version Code
1.0.0	100
1.1.0	110
1.2.0	120
1.3.0	130
1.3.1	131

[Note]

- No part of this manual may be copied or reproduced in whole or in part without the written permission of AIOI·SYSTEMS CO., LTD.

- The **marblePORT** specifications and the contents of this manual may be revised without prior notice.

  ∗ Android is a trademark of Google Inc.
  ∗ Eclipse is a trademark of Eclipse Foundation, Inc.
  ∗ Java is trademarks or registered trademark of Oracle Corporation.
  ∗ Osaifu-Keitai is a trademark or registered trademark of NTT DOCOMO, INC.
  ∗ FeliCa is a registered trademark of Sony Corporation.

marblePORT Ver.1.3
Programming Guide

AIOI·SYSTEMS CO., LTD.

WiRa Oomori Bldg, 8F
1-6-8 Oomori Kita, Ota-ku, Tokyo 143-0016 JAPAN
Tel: 03-3764-0228
Fax: 03-3764-7520
e-mail: info@hello-aioi.com
web: http://www.hello-aioi.com/