

# QSS Network Service

- Version 2.3.0.0 -

## Revision History

Revision Date	Description
Sept. 05, 2001	Newly created
Sept. 25, 2001	Release version 1.0.2.  Member variable Version in QSS_PRINTER_INFO structure is changed from unsigned short to unsigned long. Description of QssGetOrderState is corrected.  The range of WithBorderC, WithBorderP, and WithBorderH in QSS_ORDER_PARAM structure is defined to 0-99, and the unit is changed to 1/10 mm. QSS_INVALID_PAPERLENGTH is added as the return value for QssSetOrder function.  Values to be set to member variable QssState in QSS_PRINTER_STATE structure is corrected.  Description for member variable TotalPrintNum in QSS_PRINTER_STATE structure is added.  In "1-3-1 Functions", information on QSS process solution to be obtained is changed.  Member variable RemaindQuantityXXX in QSS_PRINTER_STATE structure is changed to Unused.  "17: Photo Shop Document" is added as the bit assignment for member variable SupportImageFormat in QSS_PRINTER_STATE structure.  Descriptions for member variables CvpString1 and CvpString2 in QSS_FRAME_PARAM structure are added.
Oct. 30, 2001	Release version 1.0.3.  Information on printable image format and QSS spool area is added to "1-3-2 Limitations".  Return values are added to QssTransmitFile function.  Changes are made to the order status that can be acquired with QssGetOrderState function.  QssGetProfile function and its accompanying structures, QSS_PROFILE_INFO and QSS_PROFILE_PIPE are added in order to get the profile. QssLookup function is added.  Member variables PaperLengthMin and PaperLengthMax are added to QSS_PAPER_INFO structure.  Change, addition and description are added to member variables for QSS_PRINT_CHANNEL structure.  Changes are made to member variables RepeatNum and RepeatPos in QSS_FRAME_PARAM structure.  Member variable CmsFlg is added to QSS_ORDER_PARAM structure. Also, descriptions are added to member variables PaperWidth, Surface, IndexPaperWidth, IndexSurface, PaperLengthC, PaperLengthP, PaperLengthH, and IndexPrintFlg. SpoolerSpace is added to QSS_PRINTER_STATE structure. Descriptions for member variables QssState and TotalPrintNum are changed. NOTE is added to member variables NameC, NameP, and NameH in QSS_PU_INFO structure.  Reserved areas are secured for all the structures.  Input/output is added to member variables in all the structures.
Nov. 08, 2001	QSS is added to the trademark notice.
Dec. 10, 2001	The term "Order number" is changed to "Request number".
Dec. 18, 2001	Description is added to QSS_PU_INFO structure.

	<p>A note regarding single-magazine type QSS is added to the description of QSS_ORDER_PARAM structure.</p> <p>The term "Order number" is changed to "Request number".</p>
Sept. 2, 2002	<p>Release version 1.0.4</p> <p>Member variables PaperWidth and PaperLength, Surface are added to QSS_FRAME_PARAM structure.</p>
Sept. 27, 2002	<p>Reference number (RefId) is added to QSS_FRAME_PARAM, QSS_ORDER_PARAM, and QSS_ORDER_STATE structures.</p> <p>QssAbortRefID, API that enables to cancel orders based on reference number, is added.</p>
Nov. 13, 2002	<p>Release version 1.0.5</p> <p>Two items are added to order status that can be got with QssGetOrderState function. Δ2</p> <p>API, QssGetOrderStateRefId, which is capable of getting order status based on reference number is added. Δ2</p> <p>API, QssGetOrderHistory, which is capable of getting order history is added. Δ2</p> <p>IPAddress, Port, Version, and Level are added to QSS_CLIENT_INFO structure. Δ2</p> <p>QSS_ORDER_PRINTED and QSS_ORDER_CANCELED are added to OrderState of QSS_ORDER_STATE structure. Δ2</p>
Nov. 26, 2002	<p>IPAddress is added to QSS_FRAME_PARAM structure. Δ3</p> <p>Values to be set to PrintSize of QSS_FRAME_PARAM structure are changed. Δ3</p> <p>PaperLength is available in QSS_FRAME_PARAM structure. Δ3</p> <p>QSS_ORDER_STATE structure is now in the original state, and QSS_ORDER_STATE_EX structure is added instead. Δ3</p> <p>Parameters are changed for QssGetOrderStateRefId function. Δ3</p>
Dec. 25, 2002	<p>Allowable ranges were defined to BufferNum parameter of QssGetOrderState API and QssGetOrderStateRefId API. This is because there are cases where API does not function correctly due to the restriction of RPC when a value that exceeds 10000 is defined for BufferNum. Δ4</p>
Oct. 30, 2003	<p>Version 1.0.6 was released.</p> <p>SorterNum was added as a member to QSS_ORDER_PARAM structure. Δ5</p>
Nov. 12, 2003	<p>Allowable ranges were defined to OrderNo parameter of QssAbort and QssSetPUInfo functions. Δ6</p> <p>Allowable range was defined to RefId parameter of QssAbortRefId function. Δ6</p> <p>Allowable ranges were defined to OrderNo of QSS_FRAME_PARAM, QSS_ORDER_PARAM, and QSS_ORDER_STATE structures. Δ6</p> <p>Allowable range was defined to RefId of QSS_ORDER_STATE_EX structure. Δ6</p>
Mar. 23, 2004	<p>QSS_RECEIVE_ABORT was added to the return value of QssSetOrder function. Δ7</p> <p>QSS-32 and QSS-33 were included in the models that support IndexPrintFlag of QSS_ORDER_PARAM structure. Δ7</p> <p>QSS_INDEX_CD40, QSS_INDEX_CD40A, QSS_INDEX_CD40B, QSS_INDEX_3WL, and QSS_INDEX_3WL_18 were added to IndexPrintFlag of QSS_ORDER_PARAM structure. Δ7</p> <p>A description was added to describe that QSS-30 does not support SorterNum of QSS_ORDER_PARAM structure. Δ7</p> <p>QSS_MAGAZINE_C was added to MagazineState of QSS_PAPER_INFO structure. Δ7</p> <p>QSS_PRINTTYPE_LONG was added to PrintType of QSS_PRINT_CHANNEL structure. Δ7</p> <p>QSS_INPMEDIA_CTERM, QSS_INPMEDIA_RDS, QSS_INPMEDIA_SD, QSS_INPMEDIA_MS,</p>

	<p>QSS_INPMEDIA_STORAGE, and QSS_INPMEDIA_USB were added to InpMediaType of QSS_PRINT_CHANNEL structure. Δ7</p> <p>QSS_INDEX_CD40, QSS_INDEX_CD40A, QSS_INDEX_CD40B, QSS_INDEX_3WL, and QSS_INDEX_3WL_18 were added to IDPSize of QSS_PRINT_CHANNEL structure. Δ7</p> <p>QSS_OUTPMEDIA_SD, QSS_OUTPMEDIA_MS, QSS_OUTPMEDIA_BRAVO, and QSS_OUTPMEDIA_USB were added to OutMediaSw of QSS_PRINT_CHANNEL structure. Δ7</p>
April 26, 2004	<p>QSS spool area can now store the data up to 999 GByte instead of 3 Gbyte. Δ8</p> <p>QSS_INDEX_4WL_18 was added to IndexPrintFlag of QSS_ORDER_PARAM structure. Δ8</p> <p>QSS_INDEX_4WL_18 was added to IDPSize of QSS_PRINT_CHANNEL structure. Δ8</p>
April 27, 2004	<p>Version 2.0.0 was released.</p> <p>Description that it is also possible to start printing as soon as the print data transfer from Client to QSS is completed was added in "1-4-1. Print sequence" as well as the basic print sequence for this case. Δ9</p> <p>QssTransmitFile2 and QssSetOrder2 API's were added so that it is possible to start printing as soon as print data transfer from Client to QSS is completed. Δ9</p>
Aug. 03, 2004	<p>Return values for QssTransmitFile, QssTransmitFile2, and QssSetOrder2 were corrected. Δ10</p> <p>Reasons are:</p> <ul style="list-style-type: none"> <li>- NetOrder print is now available in not only NetOrder mode but also normal mode. Δ10-1, 10-4</li> <li>- When image file is found to be illegal with the first print, an error will be returned to the Client as with R2R. Δ10-2, Δ10-3</li> </ul>
Sept. 06, 2004	<p>Description of "Fast Print" was added. Δ12-1</p> <p>A description was added to QssTransmitFile2 and QssSetOrder2. Δ12-2, Δ12-3</p> <p>Description of CvpString1 and CvpString2 were corrected. Δ12-4 and Δ12-11</p> <p>QSS_INPMEDIA_XD_CARD, QSS_INPMEDIA_MINI_SD, and QSS_INPMEDIA_MS_DUO were added to InpMediaType of QSS_PRINT_CHANNEL structure. Δ12-5, Δ12-6, Δ12-7</p> <p>QSS_OUTPMEDIA_XD_CARD, QSS_OUTPMEDIA_MINI_SD, and QSS_OUTPMEDIA_MS_DUO were added to OutMediaSw of QSS_PRINT_CHANNEL structure. Δ12-8, Δ12-9, Δ12-10</p> <p>Noritsu Character Code Tables were added. Δ12-12</p>
Dec.14, 2004	<p>Index R12 was added. Δ13-1, Δ13-2</p> <p>The following variables were added to QSS_PRINTER_STATE.</p> <ol style="list-style-type: none"> <li>1. IsNetOrderMode Δ13-3</li> <li>2. IsCalibrationMode Δ13-4</li> </ol> <p>The maximum possible value of FrameNum and FrameNo for QSS_FRAME_PARAM2 was extended from 999 to 9999. Δ13-5, Δ13-6</p> <p>The maximum possible value of FrameNum for QSS_ORDER_PARAM2 was extended from 999 to 9999. Δ13-7</p>
Feb. 04, 2005	<p>QSS_INPMEDIA_DVD_ROM was added to the InpMediaType of QSS_PRINT_CHANNEL structure. Δ14-1</p> <p>QSS_OUTPMEDIA_DVD_ROM was added to the OutMediaSw of QSS_PRINT_CHANNEL structure. Δ14-2</p>
Feb. 04, 2005	<p>Rotate of the QSS_FRAME_PARAM2 structure is now available to allow every input image to be rotated. Δ15-1, Δ15-7</p> <p>The following variables were added to the QSS_FRAME_PARAM2 structure to allow every input image to be cropped.</p> <ol style="list-style-type: none"> <li>1. TrimStartPointX Δ15-2, Δ15-8</li> </ol>

	<ol style="list-style-type: none"> <li>2. TrimStartPointY <math>\Delta</math>15-3, <math>\Delta</math>15-9</li> <li>3. TrimSizeX <math>\Delta</math>15-4, <math>\Delta</math>15-10</li> <li>4. TrimSizeY <math>\Delta</math>15-5, <math>\Delta</math>15-11</li> <li>5. TrimUnitSize <math>\Delta</math>15-6, <math>\Delta</math>15-12</li> </ol>
Feb. 05, 2005	<p>The following variables were added to the QSS_ORDER_PARAM2 structure.</p> <ol style="list-style-type: none"> <li>1. IndexPrintNum <math>\Delta</math>16-1, <math>\Delta</math>16-16</li> <li>2. OutMediaFlg <math>\Delta</math>16-2, <math>\Delta</math>16-17</li> <li>3. OutMediaFormat <math>\Delta</math>16-3, <math>\Delta</math>16-18</li> <li>4. OutMediaNum <math>\Delta</math>16-4, <math>\Delta</math>16-19</li> <li>5. OutMediaQualityType <math>\Delta</math>16-5, <math>\Delta</math>16-20</li> <li>6. OutMediaQuality <math>\Delta</math>16-6, <math>\Delta</math>16-21</li> <li>7. OutMediaSize <math>\Delta</math>16-7, <math>\Delta</math>16-22</li> <li>8. OutMediaViewer <math>\Delta</math>16-8, <math>\Delta</math>16-23</li> <li>9. LabelIndexPrintFlg <math>\Delta</math>16-9, <math>\Delta</math>16-24</li> <li>10. LabelIndexPrintNum <math>\Delta</math>16-10, <math>\Delta</math>16-25</li> <li>11. LabelIndexPaperWidth <math>\Delta</math>16-11, <math>\Delta</math>16-26</li> <li>12. LabelIndexSurface <math>\Delta</math>16-12, <math>\Delta</math>16-27</li> <li>13. Priority <math>\Delta</math>16-13, <math>\Delta</math>16-28</li> <li>14. PrintMode <math>\Delta</math>16-14, <math>\Delta</math>16-29</li> <li>15. Wait <math>\Delta</math>16-15, <math>\Delta</math>16-30</li> </ol>
Feb. 05, 2005	<p>FinishTime was added to the QSS_ORDER_STATE structure. <math>\Delta</math>17-1, <math>\Delta</math>17-2</p> <p>FinishTime was added to the QSS_ORDER_STATE_EX structure. <math>\Delta</math>17-3, <math>\Delta</math>17-4</p>
Feb. 07, 2005	<p>QSS_INVALID_OUTMEDIA_PARAM was added as a return value of QssSetOrder2. <math>\Delta</math>18-1</p> <p>SystemInfo was added to the QSS_PRINTER_INFO structure. <math>\Delta</math>18-2, <math>\Delta</math>18-3</p> <p>EnableOutMediaViewer was added to the QSS_PRINTER_STATE structure. <math>\Delta</math>18-4, <math>\Delta</math>18-5</p> <p>SystemInfo was added to the QSS_PRINTER_ENUM structure. <math>\Delta</math>18-6, <math>\Delta</math>18-7</p> <p>WithBorder of QSS_FRAME_PARAM2 structure is now available. <math>\Delta</math>18-8, <math>\Delta</math>18-10</p> <p>Save was added to the QSS_FRAME_PARAM2 structure. <math>\Delta</math>18-9, <math>\Delta</math>18-11</p>
Mar. 23, 2005	<p>Description was added to the QssSetPulInfo function. <math>\Delta</math>19-1</p> <p>Description of Status of the QSS_ORDER_HISTORY structure was corrected.</p> <p>QSS_ORDER_STATUS_PRINTED -&gt; QSS_ORDER_PRINTED <math>\Delta</math>19-2</p> <p>QSS_ORDER_STATUS_CANCELED -&gt; QSS_ORDER_NONE <math>\Delta</math>19-3</p>
Apr. 18, 2005	<p>QSS_INVALID_PARAMETER was added as a return value of QssTransmitFile2. <math>\Delta</math>20-1</p> <p>Descriptions were added to the following values of the QSS_ORDER_PARAM2 structure:</p> <p>IndexPrintNum <math>\Delta</math>20-2</p> <p>OutMediaQualityType <math>\Delta</math>20-3</p> <p>OutMediaQuality <math>\Delta</math>20-4</p> <p>OutMediaViewer <math>\Delta</math>20-5</p> <p>LabelIndexPrintFlg <math>\Delta</math>20-6</p> <p>Priority <math>\Delta</math>20-7</p>
May 23, 2005	<p>EnablePriority was added to the QSS_ORDER_PARAM2 structure. <math>\Delta</math>21</p>
Jun. 08, 2005	<p>Possible value for QSS_PRIORITY_NORMAL of Priority of the QSS_ORDER_PARAM2 structure was extended.</p> <p>150 – 249 -&gt; 150 – 9999 <math>\Delta</math>22-1</p>

	<p>The following values were added to Priority of the QSS_ORDER_PARAM2 structure.</p> <p>QSS_PRIORITY_LOW Δ22-2</p> <p>QSS_PRIORITY_NONE Δ22-3</p>
Oct. 17, 2005	<p>In the QSS_FRAME_PARAM structure:</p> <ul style="list-style-type: none"> <li>- WithBorder was made available. Δ23-1, Δ23-6</li> <li>- PaperFittingFlg was made available. Δ23-2, Δ23-7</li> <li>- Way was added. Δ23-3, Δ23-8</li> <li>- Reserve2 was added. Δ23-4, Δ23-9</li> <li>- EnablePaperFittingFlg was added. Δ23-5, Δ23-10</li> </ul> <p>In the QSS_FRAME_PARAM2 structure:</p> <ul style="list-style-type: none"> <li>- PaperFittingFlg was made available. Δ23-11, Δ23-13</li> <li>- EnablePaperFittingFlg was added. Δ23-12, Δ23-14</li> </ul>
Oct. 17, 2005	2.1.0 release
Jan. 25, 2006	<p>In the QSS_PAPER_INFO structure:</p> <ul style="list-style-type: none"> <li>- QSS_MAGAZINE_A2 was added as the setting value of Magazine State. Δ24-1</li> <li>- Description of Paper Remaind was corrected. Δ24-2</li> </ul> <p>In the QSS_ORDER_PARAM2 structure:</p> <ul style="list-style-type: none"> <li>- PaperWidthD and SurfaceD were added. Δ24-3, Δ24-4, Δ24-5, Δ24-6, Δ24-7, Δ24-8, Δ24-9</li> <li>- FrontPrintString and FrontPrintFlg were added. Δ24-10, Δ24-11, Δ24-12, Δ24-13, Δ24-14</li> <li>- Comment was made available. Δ24-15, Δ24-16</li> </ul>
Jan. 25, 2006	2.2.0 release
Feb. 14, 2006	<p>Description of FrontPrintString of the QSS_FRAME_PARAM2 structure was corrected. Δ25-1</p> <p>“19 characters can be set at maximum” -&gt; “31 characters can be set at maximum”</p>
Feb. 14, 2006	2.2.0 2 <sup>nd</sup> release
Jan. 5, 2007	<p>Allowable range of values for RepeatNum was extended for the QSS_FRAME_PARAM structure. Δ26-1</p> <p>Allowable range of values for RepeatNum was extended for the QSS_FRAME_PARAM2 structure. Δ26-2</p> <p>2.3.0 release</p>

## &lt; Table of contents &gt;

Introduction .....	9
Chapter 1. Overview .....	10
1-1 Outline of QSS Network Service.....	10
1-1-1 Outline .....	10
1-1-2 System Configuration.....	10
1-1-3 Connection .....	10
1-1-4 Communication.....	10
1-1-5 Supported Platform.....	10
1-1-6 Scope of Software Development.....	11
1-2 Outline of QSS .....	13
1-2-1 Outline .....	13
1-2-2 Features.....	13
1-2-3 Machine Configuration .....	13
1-2-4 Startup checks.....	14
1-2-5 Printing operation .....	14
1-2-6 Closedown checks .....	15
1-2-7 Information required to auto-print on QSS.....	15
1-3 Network Service.....	17
1-3-1 Functions .....	17
1-3-2 Limitations .....	17
1-4 Network Print Flow.....	19
1-4-1 Print sequence .....	19
1-4-2 Fast Print $\Delta$ 12-1.....	20
1-5 Build the Client Application.....	23
1-5-1 Interface implementation.....	23
1-5-2 Sample code .....	25
Chapter 2. Application Programming Interface Reference.....	34
QssGetName.....	35
QssTransmitFile.....	35
QssSetOrder.....	36
QssAbort.....	36
QssSetPUInfo.....	37
QssGetPaper.....	38
QssGetError.....	39
QssGetOrderState .....	40
QssGetPrinterState .....	41
QssGetChannellInfo .....	41
QssGetSumInfo.....	42
QssGetProfile.....	42
QssLookup.....	43
QssAbortRefId $\Delta$ 1.....	43
QssGetOrderStateRefId $\Delta$ 2.....	44
QssGetOrderHistory $\Delta$ 2.....	45
QssTransmitFile2 $\Delta$ 9.....	46
QssSetOrder2 $\Delta$ 9.....	47
QSS_PRINTER_INFO structure .....	48
QSS_CLIENT_INFO structure.....	49
QSS_FRAME_PIPE structure .....	49

QSS_FRAME_PARAM structure .....	50
QSS_ORDER_PARAM structure .....	53
QSS_PAPER_INFO structure .....	55
QSS_ERROR_INFO structure .....	56
QSS_ORDER_STATE structure .....	57
QSS_ORDER_STATE_EX structure Δ3 .....	57
QSS_PRINTER_STATE structure .....	58
QSS_PRINT_CHANNEL structure .....	60
QSS_PU_INFO structure .....	64
QSS_SUM_INFO structure .....	65
QSS_PROFILE_INFO structure .....	67
QSS_PROFILE_PIPE structure .....	67
QSS_PRINTER_ENUM structure .....	68
QSS_DATETIME structure Δ2 .....	68
QSS_ORDER_HISTORY structure Δ2 .....	69
QSS_FRAME_PARAM2 structure Δ9 .....	71
QSS_ORDER_PARAM2 structure Δ9 .....	74
Appendix 1: inch – 1/10mm Conversion Table .....	80
Appendix 2: Noritsu Character Code Table Δ12-12 .....	81
Glossary .....	83
Trademarks .....	86

## Introduction

This document describes the interface between the external terminal such as server and QSS (28, 29 and 30 series). Please note the description in this document is applicable to auto-print function.

## Chapter 1. Overview

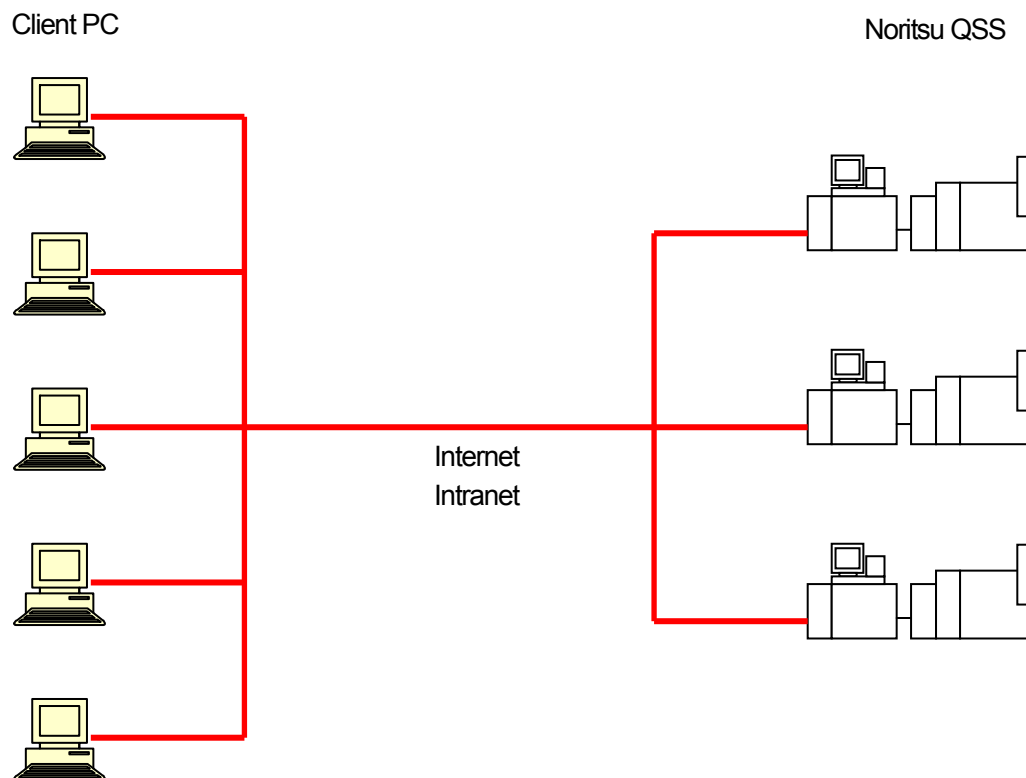
### 1-1 Outline of QSS Network Service

#### 1-1-1 Outline

QSS Network Service provides the function to perform a series of print operation (auto-print) on QSS-28, 29 and 30 series from the external terminal such as server.

#### 1-1-2 System Configuration

The red line in the figure below is the interface with QSS.



#### 1-1-3 Connection

External terminal such as Client PC ("Client") and QSS are connected by using Ethernet. Multiple Clients can be connected with multiple QSS's.

#### 1-1-4 Communication

Communication between Client and QSS can be achieved by using RPC (Remote Procedure Call) via TCP/IP.

#### 1-1-5 Supported Platform

Currently, QSS Network system supports the following OS:

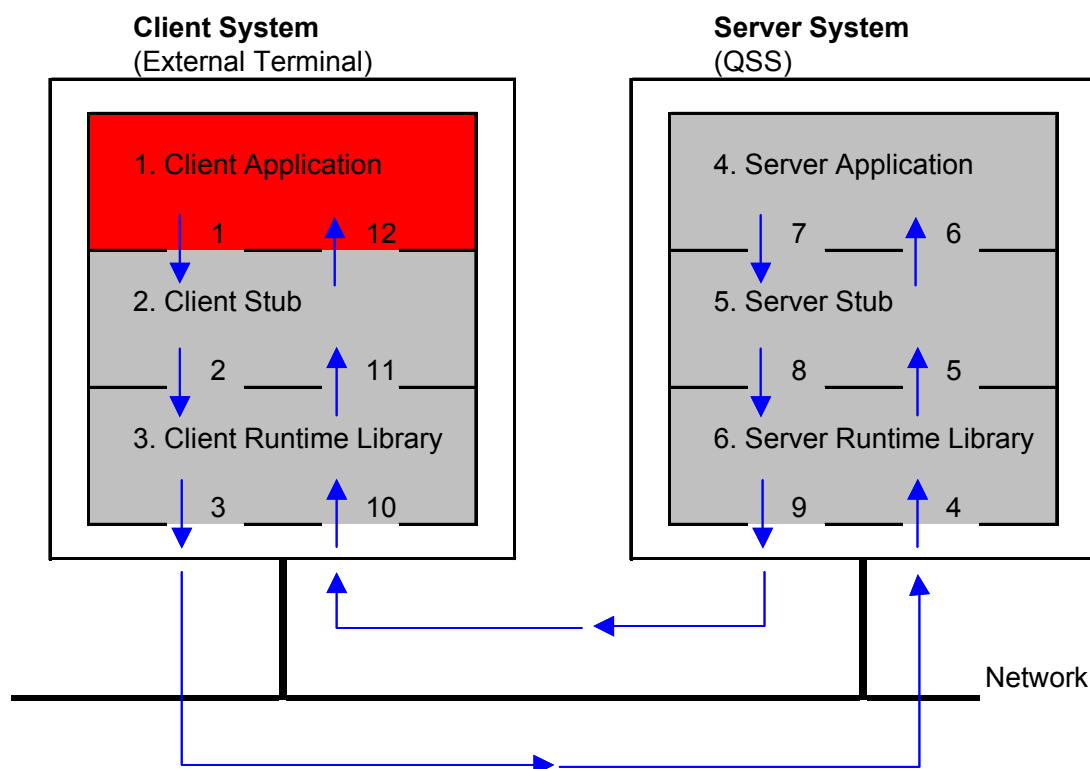
- Windows 98SE, Windows Me, Windows NT 4.0 Workstation (SP4 or up),
- Windows 2000 Professional (SP2 or up)

Whether the following will be supported or not is TBD:

- Macintosh
- Linux

### 1-1-6 Scope of Software Development

Scope of Software Development is as follows:



Item	Developed by
1 Client Application (Shop Server Application)	Client Developer
2 Client Stub	NKC
3 Client Runtime Library (RPC Runtime) To be prepared by Client, if required. Standard implementation in Windows 2000	-
4 Server Application (QSS Application)	NKC
5 Server Stub	NKC
6 Server Runtime Library (RPC Runtime) Standard implementation in Windows 2000	-

#### < RPC Process Flow >

- 1) Client Application calls Client Stub on Local system.
- 2) Client Stub converts the data transferred from Client Application to NDR format so it can transfer the data to Server.
- 3) Client Stub then calls the function in the Client Runtime Library to look for the RPC request and destination of the data (Server Application), and send it to the destination.
- 4) Function in the Server Runtime Library receives PRC request and data transferred from the Client.
- 5) The data received by Server is transferred to Server Stub and restored.
- 6) Server Stub calls the actual procedure in Server Application by using the restored data.
- 7) Data obtained by executing remote procedure is transferred to Server Stub again.
- 8) Server Stub converts the data obtained to NDR format in order to transfer it to Client System in the same way as Client Stub has done.

- 9) Server Stub calls the function in the Server Runtime Library to transfer the data to Client.
- 10) Function in Client Runtime Library receives the data returned from the Server.
- 11) Data is transferred to the Client Stub and then restored.
- 12) The Client Stub writes the data restored into the memory in the Client Application, and close RPC process.

< Compatibility >

Currently, OS of the Server (QSS-28, 29, and 30 series) is Windows 2000 Professional.

Microsoft Remote Procedure Call (Microsoft RPC) implemented in Windows 98, NT, 2000, etc. is compatible with RPC functions that complies with DCE (Distributed Computing Environment) of OSF (Open Software Foundation).

Besides, there are other RPC's such as ONC (Open Network Computing) of Sun Microsystems.

ONC RPC is capable of using this RPC on the OS generally called UNIX.

There is no compatibility between DCE and ONC.

**Therefore, it is required for Client System to use RPC runtime that complies with the RPC of OSF DCE.**

## 1-2 Outline of QSS

### 1-2-1 Outline

QSS is the acronym of Quick Service System that is a system capable of printing and enlargement from the films developed with mini-lab.

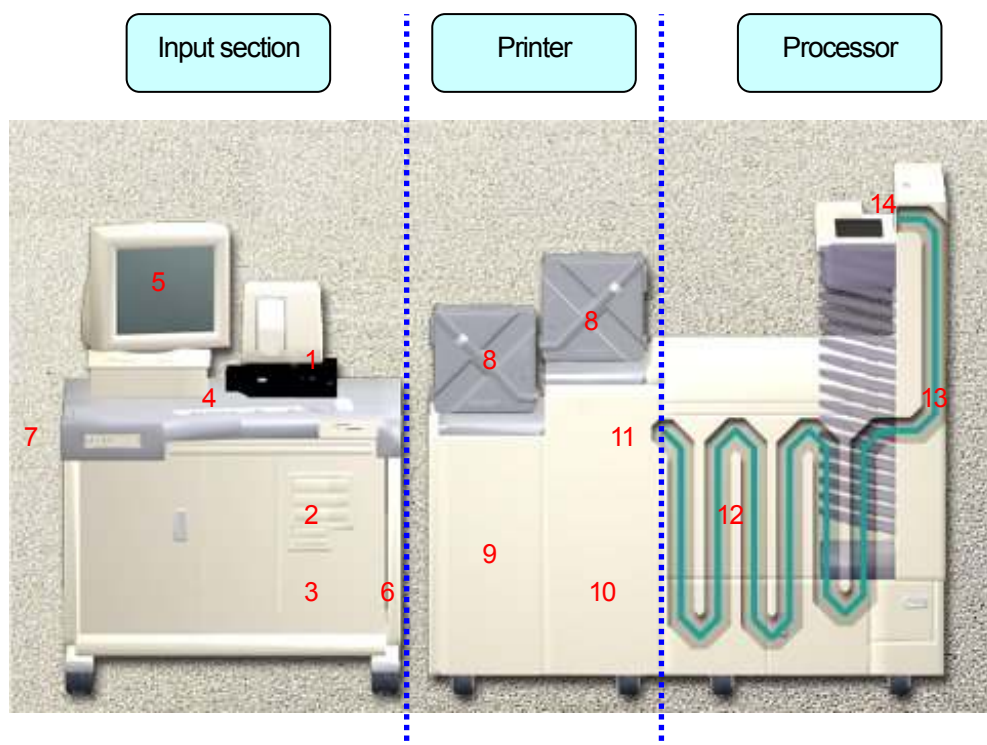
Some QSS's such as QSS-28, 29, and 30 series are capable of digital printing.

### 1-2-2 Features

- Full digital process from scanning of film to printing
- Employ sophisticated image correction
- High-speed digital processing of image data
- Capable of inputting image data from storage media
- Easy and beautiful printing directly from digital camera
- Capable of exposing the high-quality image onto silver halide paper, which can save cost
- High-speed print operation
- Capable of printing the printing information on the back of print
- Equipped with order classification device
- Equipped with auto-tank-cleaning and auto-water-refilling system as the standard equipment
- Equipped with auto-replenisher system and effluent tank as standard equipment

### 1-2-3 Machine Configuration

As shown in the figure below, a QSS can be separated into three (3) sections.



#### (1) Input section

Purpose:

QSS operator scans film, and/or reads image data from a storage media such as CD-ROM, FD, Smart Media, etc. and edits image to be printed, and/or specifies the number of copies to be made by using the keyboard and/or mouse while looking at the display monitor.

Functions:

1. Scan film

2. Read image data from storage media
3. Receive image data via network
4. Register and/or change the information needed to make prints
5. Edit, make correction to, specify the number of copies to be made of the image to be printed
6. Send print data to printer unit
7. Issue pricing sheet

## (2) Printer section

Purpose:

Expose the image data that the operator has determined to print on the input section onto the silver halide paper.

Functions:

8. Load paper from the paper magazine into the printer section
9. Expose the image onto the silver halide paper
10. Print CVP
11. Advance the exposed paper to the paper processor section

## (3) Paper Processor

Purpose:

Pass the exposed paper through each processing tank – Color Developer (CD), Bleach Fix (BF), Stabilizer (STB) –, dry and then output the paper (resultant prints) onto the paper sorter unit.

Functions:

12. Pass the exposed paper through each processing rack
13. Dry the processed paper
14. Output the paper (resultant prints) onto the paper sorter unit

### 1-2-4 Startup checks

Below is the list of startup check items.

- 1) Check the condition of the paper processor.
- 2) Clean the optical parts of the scanner section and the Auto Film Carrier.
- 3) Clean the exposure unit dust prevention glass.
- 4) Refill water to compensate for the evaporation.
- 5) Process a control strip.
- 6) Carry out the daily setup or monthly setup.

### 1-2-5 Printing operation

Below is the printing operation flow when printing from the film.

- 1) Select the print channel.  
-> Select the print channel to print from the film.  
In this print channel all the information required for printing such as print size is registered.
- 2) Select the print type.  
-> Select "Normal Print".
- 3) Select the print method.  
-> Select among "PJP", "AUTO", and "PPI".
- 4) Specify the number of index prints to be made.  
-> If no index prints are to be output, there is no need to set.
- 5) Specify the type of film.  
e.g. 135F (full frame), 135H (half frame), 6 x 4.5, etc.
- 6) Set the film to the Auto Film Carrier.

-> When "AUTO" is selected for the print method, printer will start printing upon inserting film to the AFC.

- 7) Check the color and density of the image displayed on the monitor and make correction if necessary.

- 8) Execute printing.

-> When the print method is "PJP", make adequate correction to the image displayed, if necessary, and press the [YES/START] key so the exposure of the images currently displayed on the monitor will start.

When the print method is "PPI", if the operator does not make any setting or correction to the image displayed, printing starts automatically after the predetermined period of time has elapsed.

- 9) Repeat the above procedure.
- 10) Check the resultant prints by comparing them with the film.

#### 1-2-6 Closedown checks

Below is the list of closedown check items.

- 1) Display the total number of prints made on that day.  
-> The operator can check number of finished prints or other print-related data.
- 2) Save the data.  
-> Various settings that have been made to the machine can be saved to a floppy disk, if necessary.
- 3) Carry out AFC head cleaning.  
-> Required in order to read the magnetic data on IX240 (APS) films correctly.
- 4) Carry out the automatic cleaning of the processor upper guides.
- 5) Empty the effluent tank.
- 6) Check the refilling water level.  
-> Add water if the level is low.
- 7) Remove the upper guides and squeegee unit and wash them.  
-> Needed only once a week.
- 8) Clean the scanner lamp light source unit.  
-> Needed only once a week.
- 9) Turn the program timer on.

NOTE: For the detailed procedure of each operation, please refer to operation manual of each QSS model.

#### 1-2-7 Information required to auto-print on QSS

Below is the list of information required to print the image data that is output from the Client via network.

- 1) Print channel information  
The following information contained in the Print Channel Information is required:
  - Print size (paper width and length) – for C, P and H.
  - Paper surface
  - WB / BL (in case of WB, width of the border) – for C, P and H.
  - Index print enabled / disabled
  - CVP enabled / disabled

NOTE: Print channel number itself is not required.

- 2) Image data information
  - Image file name
  - Number of files
  - File number (or frame number)
  - Image format
  - Number of copies
  - The position to include print count

- CVP printing string
- 3) Order Information
  - Request number
  - Paper fitting (Overall, Cut, Real size)
  - PU information (Required when issuing a pricing sheet on PU connected to QSS)
- 4) Others
  - MAC address of Client
  - ID

## 1-3 Network Service

### 1-3-1 Functions

Network Service is capable of the following:

Below is the list of functions of Network Service.

#### < Connections >

- Connect with multiple Clients (applications) simultaneously.

#### < Printing >

- Receive printing request by order

Orders will be printed in the order they are received. Even if the printing operation of an order is not complete, QSS can accept the next order if it is ready.

NOTE: An order cannot be separated to request printing on multiple QSS's.

- Print the image retouched by Client.

Also, possible to print the image data directly loaded from a digital camera.

- Print WB print and index print.

It is not necessary for Client to prepare images to make WB print and/or index print.

#### < Cancel >

- Cancel the order you have requested.
- In the following case, QSS will automatically delete the order.
  - a. When some images in an order are missing;  
-> QSS will automatically delete the order upon receipt.
  - b. When an image is available, but order information of the image is not, or when network is disconnected while transferring the order information;  
-> If order information is not received within 10 minutes after the images have been sent to QSS, QSS will automatically delete the order.  
(Delete result will remain in log.)
  - c. QSS is shut down while transferring the order information.  
-> QSS will delete the order data while re-starting up.

#### < Obtaining information >

- Obtain the following information.
  - a. Order status
  - b. QSS status (including error and attention messages)
  - c. QSS process solution temperature.Information regarding the residual amount of solution cannot be obtained.

#### < Others >

- Output pricing sheet with Pricing Unit connected to QSS.

NOTE: The following functions must be supported by Client application, if required:

- Client receives print order even though the QSS is not ready to print.  
Upon the completion of connection with QSS, the orders stored in Client will be sent to QSS.
- In case the order transfer to QSS is failed, a message to warn the operator is shown on Client display.

### 1-3-2 Limitations

The following functions are not available:

1. Color management system

2. Notifying QSS of the error occurred on Client
3. Printing on the paper of different width within an order
4. QSS correction or retouching of the image sent from Client

**NOTE:**

- Printing will be performed in the order that the QSS receives the print data.  
Please note that the QSS can change the order of printing.
- When reception of order is prohibited on QSS, no order is accepted from Client.
- In case an error, such as paper jam, paper end, etc., arises while printing an order, the order will be canceled.
- When the paper magazine is replaced, or when the machine is recovered from an error condition, a message to ask the operator whether to feed a cut piece of the leading end of the paper or not is shown on the QSS.  
A cut piece of the leading end of the paper will not be loaded unless the operator physically presses the button on the QSS in reply to the message.
- Index prints are made in the format that the Client specifies.
- Printable image format and the size are the same as that for media printing on QSS.
- QSS spool area can store the data up to 999 Gbyte.

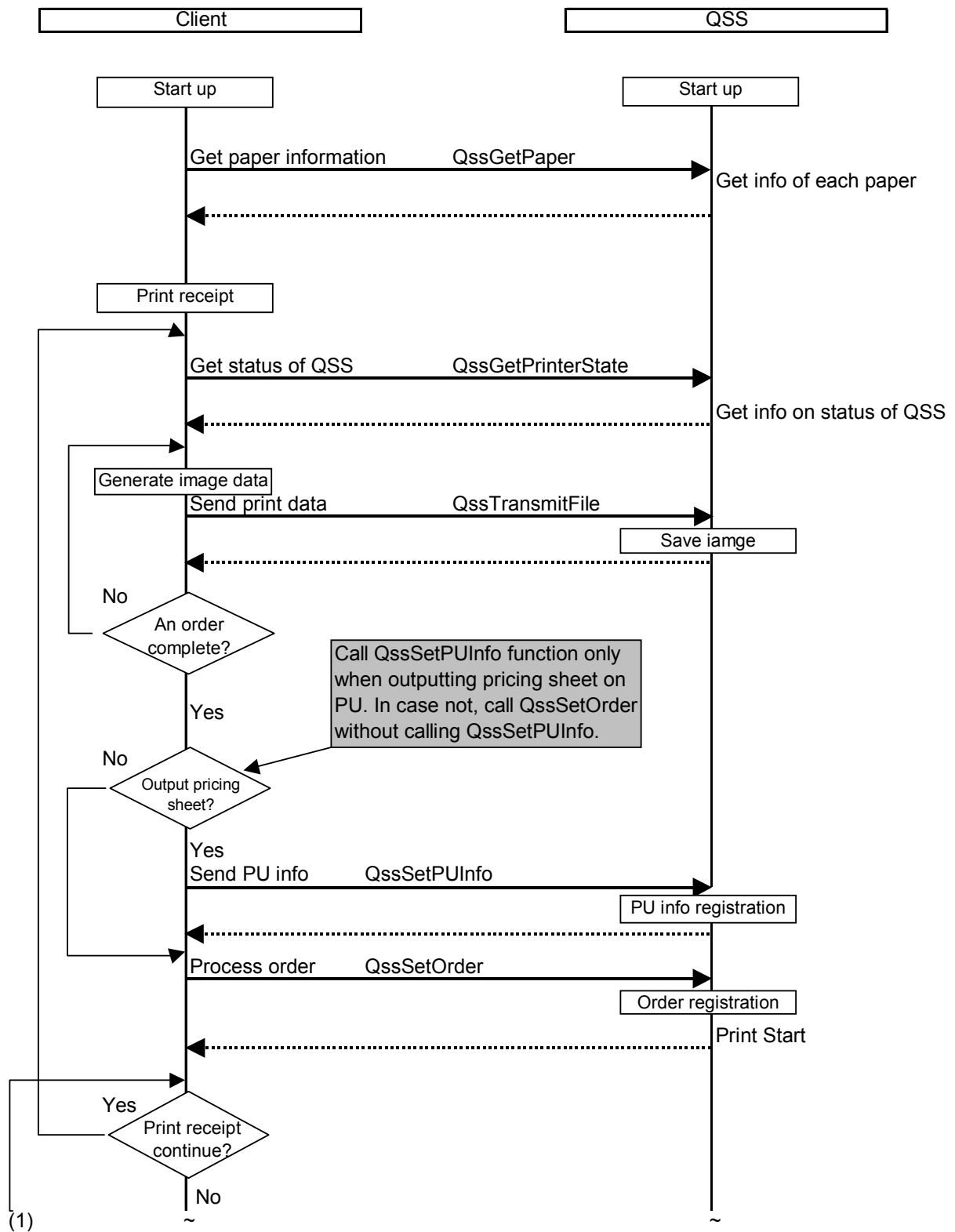
Printing will be suspended in the following case, so the adequate action has to be taken on the QSS.

- Print sorter unit on QSS is full of print.  
-> Prints need to be removed from the sorter unit.
- Solution level is low.  
-> Replenish the solution whose level is low.
- Effluent tank is full.  
-> Empty the effluent tank.
- Client requests to make print whose paper magazine is not installed on the QSS.  
-> Replace the paper magazine.
- Paper runs out.  
-> Place a new roll in the paper magazine.
- Paper gets jammed.  
-> Removed the jammed paper.
- DLP lamp is burned out.  
-> Replace the DLP lamp.
- An error that requires operator's intervention arises.  
-> Take a necessary action accordingly.

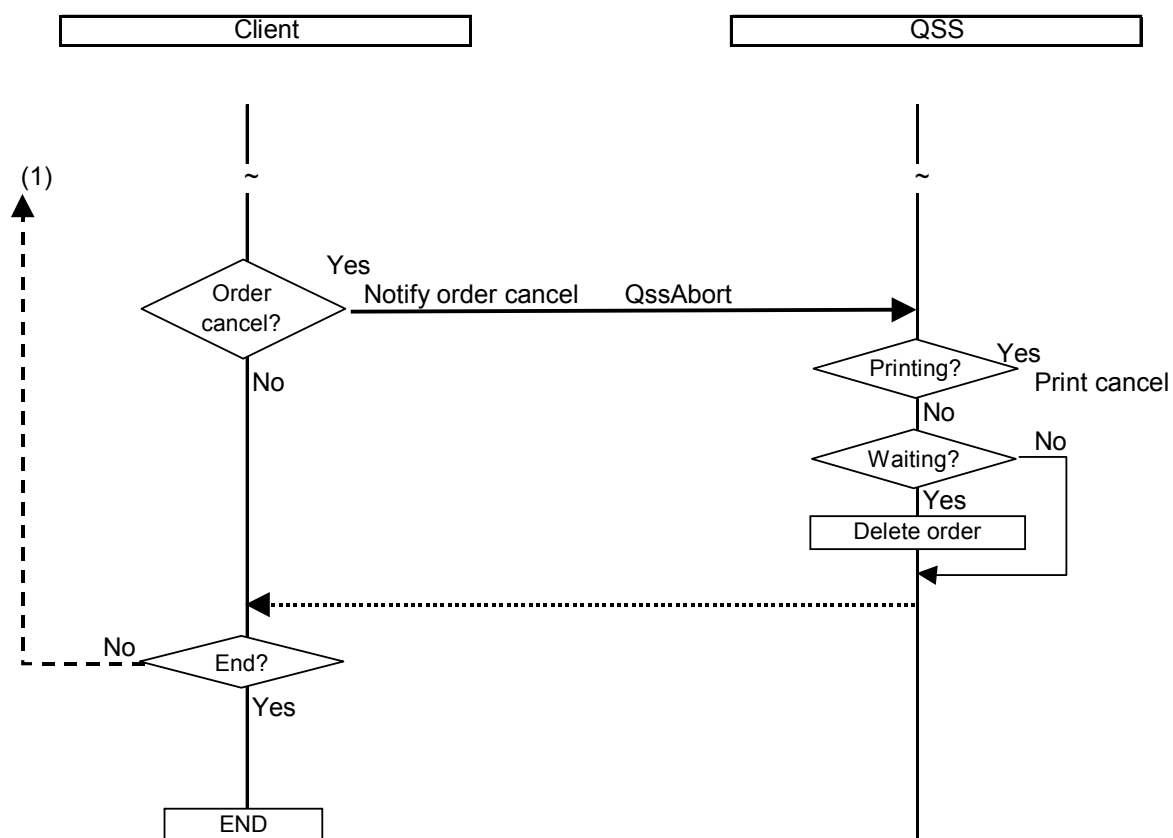
## 1-4 Network Print Flow

### 1-4-1 Print sequence

Below is the general print sequence.

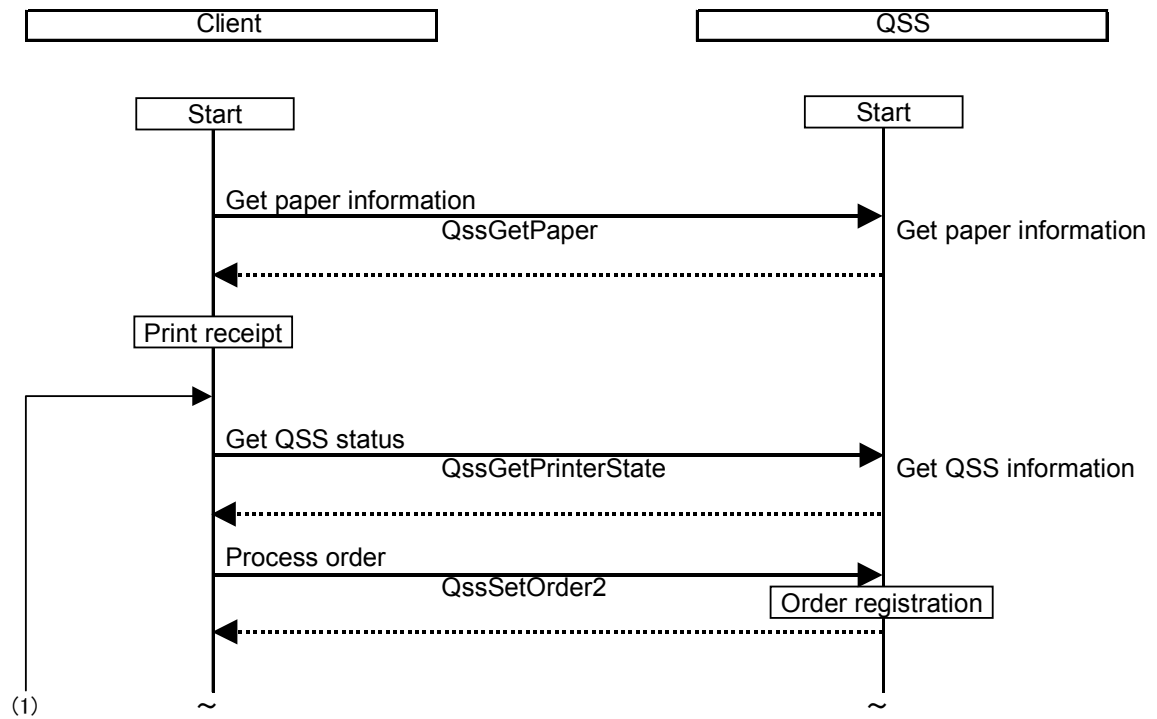


Continued from the previous page

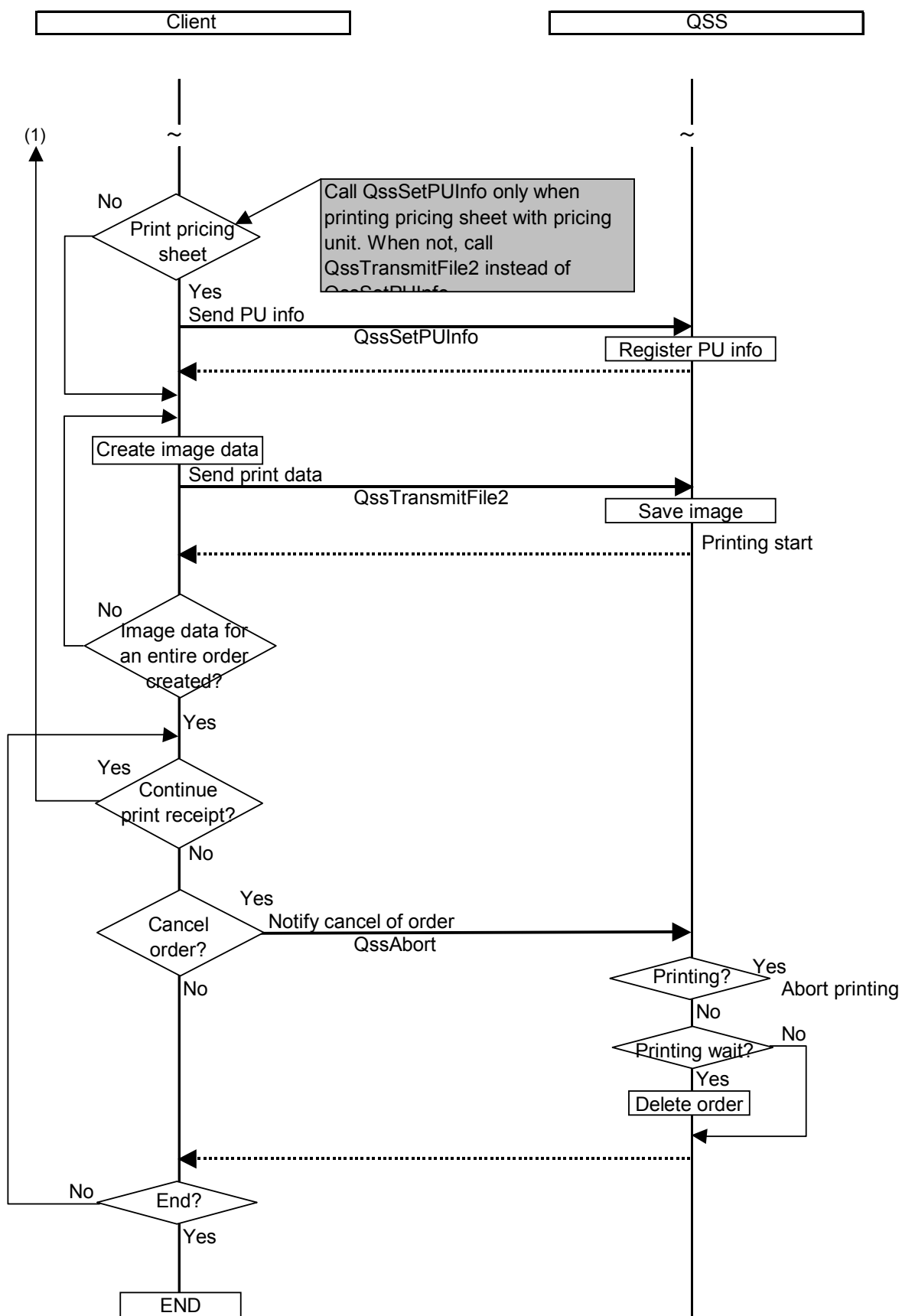
**1-4-2 Fast Print** △12-1

Fast Print function makes it possible to start printing as soon as the print data transfer from Client to QSS is completed, which will lead to higher productivity. Below illustrates the print sequence of Fast Print.

NOTE: Fast Print function is only available with NetOrder API version 2.00 and on. QSS-28, QSS-29, and QSS-30 do not have this function. △9



Continued from the previous page



## 1-5 Build the Client Application

### 1-5-1 Interface implementation

This section describes how to build the Client Application by using Microsoft Visual C++ 6.0 ("VC 6.0").

- 1) Open the target project in VC6.0.
- 2) Copy the following files that are provided to the Project folder.
  - a. QssSvr.IDL (Interface Definition File)
  - b. QssSvr.ACF (Application Configuration File)
  - c. QssDef.h (Parameter Definition File)
- 3) Add QssSvr.IDL to Workspace.
- 4) Make setting of VC6.0 project.

#### [C/C++] tab

Add "\_WIN32\_WINNT=0x500" (in case of Windows NT 4.0, "\_WIN32\_WINNT =0x400") to "Preprocessor definitions".

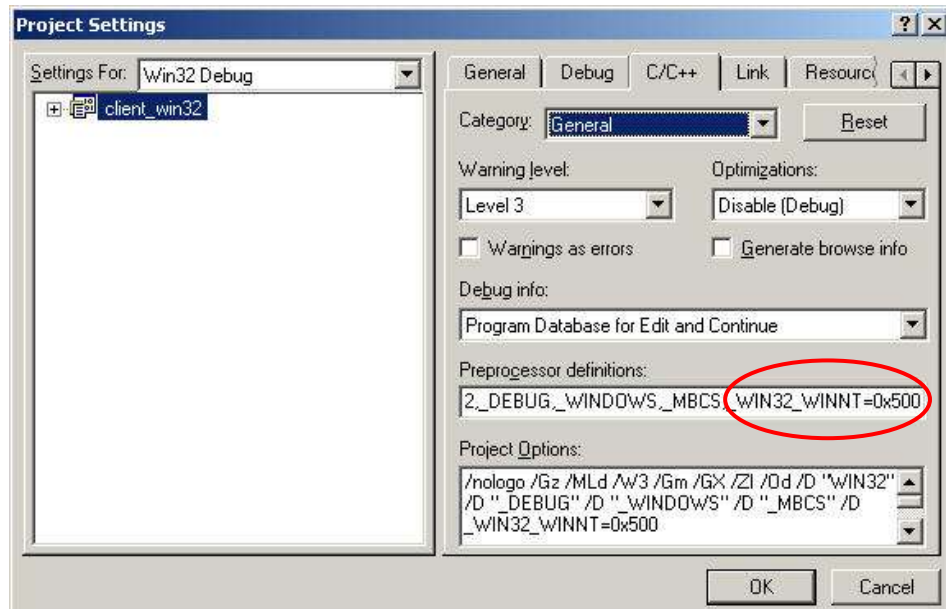


Fig. 1 Preprocessor definitions

#### [Link] tab

Add RPC libraries "rpcrt4.lib" and "rpcns4.lib" to "Object/library modules".

#### [Midl] tab

Uncheck the "MkTypLib compatible" box.

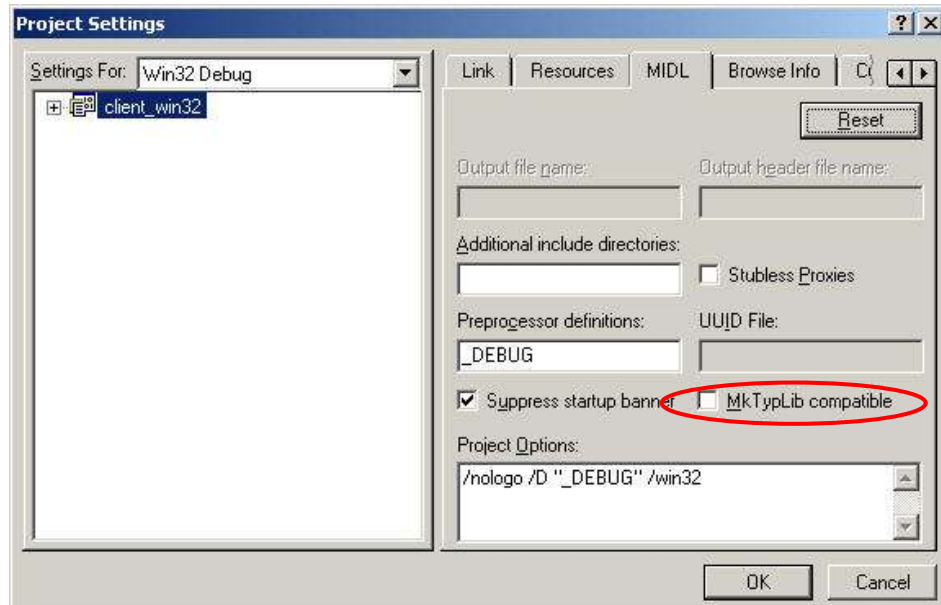


Fig. 2. Setting of "MkTypLib Compatible"

- 5) Once built, QssSvr .h and QssSvr\_c.c files are generated.  
Add QssSvr .h and QssSvr\_c.c files to workspace.
- 6) Make setting of QssSvr\_c.c file in Project Setting.

#### [C/C++] tab

Select "Not using precompiled headers" in "Precompiled Headers".

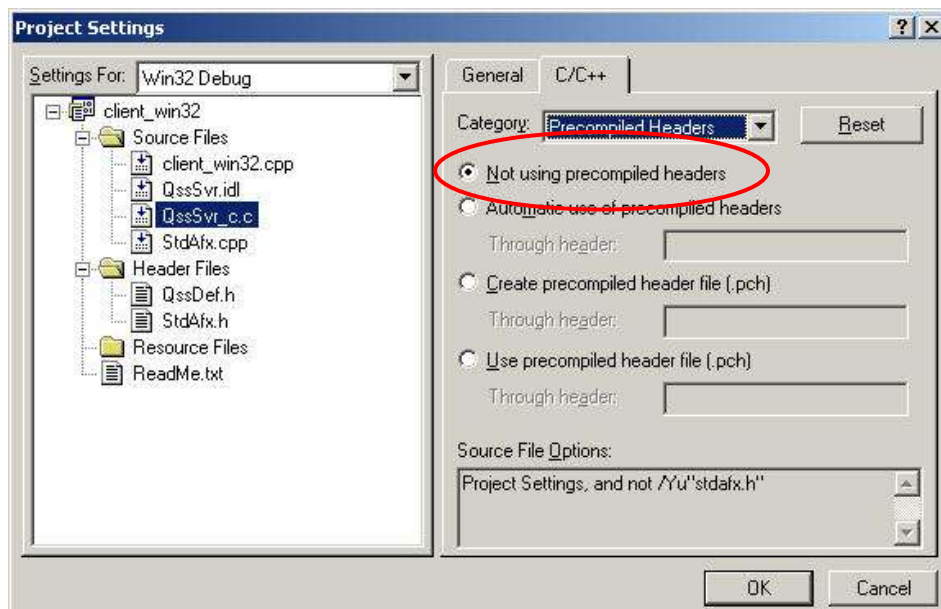


Fig. 3 Build setting of QssSvr\_c.c file

Select "\_stdcall" for "Calling convention" for "Code Generation".  
Select "8 bite \*" for "Struct member alignment."

RPC performs normally only when "\_stdcall" is selected for "Calling convention".

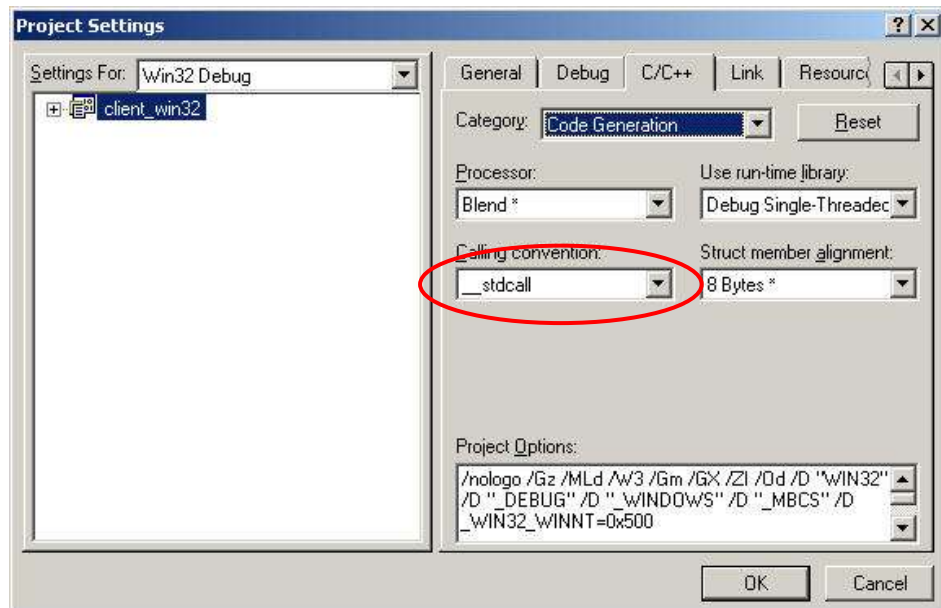


Fig. 4 Setting of code generation

7) Add the following function code to the source.

```
void __RPC_FAR * __RPC_API MIDL_user_allocate(size_t len)
{
    return (new(unsigned char [len]));
}
void __RPC_API MIDL_user_free(void __RPC_FAR * ptr)
{
    if (NULL != ptr)
        delete (ptr);
}
```

When all the settings (1-7) are complete, you are ready to use this API.

### 1-5-2 Sample code

Below is the sample code (client\_win32.cpp) on how to implement this API.

It is the sample code for registering an order to QSS and making print of the order.

The following API's are used:

QssGetName(), QssTransmitFile(), QssSetOrder(), and QssGetOrderState()

client\_win32.cpp

```
//
//                                     client_win32.CPP
//
//                                     Copyright (C) 2001 Noritsu Koki Co., Ltd. All Right Reserved
//
//
#include "stdafx.h"
#include "QssSvr.h"
#include "QssDef.h"

// Define the size of the area to be used for Pipe.
```

```

#define CLIENT_BUFFER_SIZE          2048      // Can send or receive 2048 elements

// Define the external valuable
unsigned char* g_pszStringBinding = NULL;      // Bind string of RPC
unsigned char g_pcBuffer[CLIENT_BUFFER_SIZE];  // Buffer used in Sample_PipeAlloc

// Structure to control the status of image file transfer
typedef struct _PIPE_STATE
{
    FILE*      hFile;                          // Handle of the image file to be transferred
    char*      pszFileName;                     // Name of the image file to be transferred
} PIPE_STATE;

// Function declaration
long      Sample_RpcBinding();

long      Sample_RpcUnBinding();

long      Sample_QssGetName(QSS_PRINTER_INFO __RPC_FAR *QssInfo);

long      Sample_QssTransmitFile(QSS_CLIENT_INFO ClientInfo, LPCTSTR lpszFileName,
                                QSS_FRAME_PARAM FrameParam);

long      Sample_QssSetOrder(QSS_CLIENT_INFO ClientInfo, QSS_ORDER_PARAM OrderParam);

long      Sample_QssGetOrderState(QSS_CLIENT_INFO ClientInfo, short Flag, long BufferNum,
                                long __RPC_FAR* OrderNum, QSS_ORDER_STATE __RPC_FAR *OrderState);

void      Sample_PipeAlloc(char* pcStateInfo, unsigned long nReqSize, unsigned char** pcAllocatedBuffer,
                           unsigned long* pnAllocatedSize);

void      Sample_PipePull(char* pcStateInfo, unsigned char* pcBuffer, unsigned long nMaxBufferSize,
                           unsigned long* pnSizeToSend);

int APIENTRY WinMain(HINSTANCE hInstance,
                    HINSTANCE hPrevInstance,
                    LPSTR      lpCmdLine,
                    int         nCmdShow )
{
    long      ret;                          // Return value

    //////////////////////////////////////
    // Get QSS model name and interface version
    //////////////////////////////////////

    QSS_PRINTER_INFO QssInfo;                // QSS_PRINTER_INFO structure

    // Call QssGetName.

    ret = Sample_QssGetName(&QssInfo);

    // Error handling

    //////////////////////////////////////
    // Order registration
    // Image file ("C:\Sample.JPG") transfer
    //////////////////////////////////////

    unsigned short OrderNo = 1;                // Request number
    QSS_CLIENT_INFO ClientInfo;                // QSS_CLIENT_INFO structure

```

```

// Set the client information

Istrcpy((char*)ClientInfo.User, "User");           // User name
Istrcpy((char*)ClientInfo.Host, "Host");           // Host name
ClientInfo.Address[0] = 0x01;                      // MAC address
ClientInfo.Address[1] = 0x23;
ClientInfo.Address[2] = 0x45;
ClientInfo.Address[3] = 0x67;
ClientInfo.Address[4] = 0x89;
ClientInfo.Address[5] = 0xAB;

char szFileName[MAX_PATH];                         // Image file name
QSS_FRAME_PARAM FrameParam;                       // QSS_FRAME_PARAM structure

memset(&FrameParam, 0, sizeof(QSS_FRAME_PARAM));

// Set the path name of the image file transferred
strcpy(szFileName, "C:\\Sample.JPG");

// Set the frame information
strcpy((char*)FrameParam.FileName, "Sample.JPG");   // File name
strcpy((char*)FrameParam.CvpString1, "Host User Sample.JPG"); // String to be printed on the
2nd line of the back print
FrameParam.OrderNo = OrderNo;                      // Request number
FrameParam.FrameNum = 1;                           // Number of frames
FrameParam.FrameNo = 1;                            // Frame number
FrameParam.FileSize = 84046;                        // Size of the file to be transferred (Sample.jpg)
FrameParam.ImageFormat = 0x00000001;                // Image format (JPEG)
FrameParam.PrintSize = QSS_PRINT_SIZE_C;            // Print size (Classical: 127 mm x 89
mm)
FrameParam.RepeatNum = 1;                          // Number of repeat print
FrameParam.RepeatPos = 255;                        // Repeat count include position (Not included)
FrameParam.CvpFlg = QSS_CVP_1QSS2AUX;              // CVP print setting
// (The 1st line is printed by using QSS data, and
// the 2nd line is printed by using CvpString 2.)

// Call QssTransmitFile.

ret = Sample_QssTransmitFile(ClientInfo, szFileName, FrameParam);

// Error handling (intentionally omitted)

QSS_ORDER_PARAM OrderParam;

memset(&OrderParam, 0, sizeof(QSS_ORDER_PARAM));

// Set the order information
OrderParam.OrderNo = OrderNo;                      // Request number
OrderParam.FrameNum = 1;                           // Number of frames
OrderParam.PaperWidth = 1270;                      // Paper width
OrderParam.PaperLengthC = 890;                     // Paper advance length (C)
OrderParam.PaperLengthP = 1270;                    // Paper advance length (P)
OrderParam.PaperLengthH = 1780;                    // Paper advance length (H)
OrderParam.Surface = 2;                            // Paper surface
OrderParam.WithBorderC = 4;                        // Width of white border (C)
OrderParam.WithBorderP = 4;                        // Width of white border (P)
OrderParam.WithBorderH = 4;                        // Width of white border (H)
OrderParam.IndexPrintFlg = QSS_INDEX_NONE;         // Index print
OrderParam.PaperFittingFlg = QSS_PF_CUT;           // Paper fitting

```

```

        // Call QssSetOrder

        ret = Sample_QssSetOrder(ClientInfo, OrderParam);

        // Error handling (intentionally omitted)

        //////////////////////////////////////
        // Get order status
        //////////////////////////////////////
        short Flag = 1;                // Get flag
        long BufferNum = 5;              // Buffer size
        long OrderNum = 0;              // Number of orders

        // Secure structure area to get the order status
        char* pBuffer = new char[(sizeof(QSS_ORDER_STATE) * BufferNum)];

        QSS_ORDER_STATE* pOrderState = (QSS_ORDER_STATE*)pBuffer;

        // Call QssGetOrderState.

        ret = Sample_QssGetOrderState(ClientInfo, Flag, BufferNum, &OrderNum, pOrderState);

        // Error handling (intentionally omitted)

        // Release the structure area.
        delete [] pBuffer;

        return 0;
    }

    ////////////////////////////////////// MIDL_user_allocate() //////////////////////////////////////
    void __RPC_FAR* __RPC_API MIDL_user_allocate(size_t len)
    {
        return (new(unsigned char [len]));
    }

    ////////////////////////////////////// MIDL_user_free() //////////////////////////////////////
    void __RPC_API MIDL_user_free(void __RPC_FAR * ptr)
    {
        if (NULL != ptr)
            delete (ptr);
    }

    ////////////////////////////////////// Sample_RpcBinding() //////////////////////////////////////
    long Sample_RpcBinding()
    {
        RPC_STATUS    status;                // Return value

        //////////////////////////////////////
        // Set RPC binding information
        // Communication protocol: TCP/IP
        // IP-Address: 127.0.0.1
        // Port: 5000
        //////////////////////////////////////
        unsigned char *pszUuid                = NULL;
        unsigned char *pszProtocolSequence    = (unsigned char *)"ncacn_ip_tcp";
        unsigned char *pszNetworkAddress      = (unsigned char *)"127.0.0.1";
        unsigned char *pszEndpoint            = (unsigned char *)"5000";
        unsigned char *pszOptions              = NULL;
    }

```

```

g_pszStringBinding = NULL;

// Make bind string to make bind handle.
status = RpcStringBindingCompose(
    pszUuid,
    pszProtocolSequence,
    pszNetworkAddress,
    pszEndpoint,
    pszOptions,
    &g_pszStringBinding);

switch (status)
{
case RPC_S_OK:
    break;

case RPC_S_INVALID_STRING_UUID: // The string representation of the UUID is not valid.
default:
    return -1;
}

// Make bind handle to be used for RPC communication.
status = RpcBindingFromStringBinding(
    g_pszStringBinding,
    &BindHandle);

switch (status)
{
case RPC_S_OK: // The call succeeded.
    break;

case RPC_S_INVALID_STRING_BINDING: // Invalid string binding.
case RPC_S_PROTOCOL_SEQ_NOT_SUPPORTED: // Protocol sequence not supported on this host.
case RPC_S_INVALID_RPC_PROTSEQ: // Invalid protocol sequence.
case RPC_S_INVALID_ENDPOINT_FORMAT: // Invalid endpoint format.
case RPC_S_STRING_TOO_LONG: // String too long.
case RPC_S_INVALID_NET_ADDR: // Invalid network address.
case RPC_S_INVALID_ARG: // The argument was invalid.
case RPC_S_INVALID_NAF_ID: // Invalid network address?family identifier.
default:
    return -1;
}

return 0;
}

////////// Sample_RpcUnBinding() //////////

long Sample_RpcUnBinding()
{
    RPC_STATUS    status; // Return value

    // Release the bind string.
    status = RpcStringFree( &g_pszStringBinding );

    switch (status)
    {
case RPC_S_OK: // The call succeeded.
        break;

default:
        return -1;
    }
}

```

```

        // Release the bind handle.
        status = RpcBindingFree(&BindHandle);

        switch (status)
        {
        case RPC_S_OK:                                     // The call succeeded.
            break;

        case RPC_S_INVALID_BINDING:                       // The binding handle was invalid.
        case RPC_S_WRONG_KIND_OF_BINDING:                 // This was the wrong kind of binding for the
operation.
            default:
                return -1;
        }

        return 0;
    }

    //////////// Sample_QssGetName() ////////////

    long Sample_QssGetName(
        QSS_PRINTER_INFO __RPC_FAR *QssInfo)
    {
        long    ret;                                     // Return value
        unsigned long ulCode;                             // RPC error code

        ret = Sample_RpcBinding();

        if (0 == ret)
        {
            // Procedure of exception handling: RpcTryExcept, RpcExcept, RpcEndExcept
            RpcTryExcept
            {
                ret = QssGetName(QssInfo);
            }
            RpcExcept(1)
            {
                // RPC error handling
                ulCode = RpcExceptionCode();
            }
            RpcEndExcept

            Sample_RpcUnBinding();
        }

        return ret;
    }

    //////////// Sample_QssTransmitFile() ////////////

    long Sample_QssTransmitFile(
        QSS_CLIENT_INFO      ClientInfo,
        LPCTSTR               lpzFileName,
        QSS_FRAME_PARAM FrameParam)
    {
        long    ret;                                     // Return value
        unsigned long ulCode;                             // RPC error code
        TCHAR* pszFileName;                               // Path name of the image file to be transferred
to QSS

        ret = Sample_RpcBinding();
    }

```

```

        if (0 == ret)
        {
            // Pipe setting
            QSS_FRAME_PIPE Pipe;
            PIPE_STATE PipeState;

            pszFileName = (char*)MIDL_user_allocate(sizeof(char) * (strlen(lpszFileName) + 1));

            strcpy(pszFileName, lpszFileName);
            PipeState.hFile = NULL;
            PipeState.pszFileName = (char*)pszFileName;

            Pipe.state = (char*) &PipeState;
            Pipe.alloc = Sample_PipeAlloc;
            Pipe.pull = Sample_PipePull;
            Pipe.push = NULL;

            // Procedure of exception handling: RpcTryExcept, RpcExcept, RpcEndExcept
            RpcTryExcept
            {
                ret = QssTransmitFile(ClientInfo, Pipe, FrameParam);
            }
            RpcExcept(1)
            {
                // RPC error handling
                ulCode = RpcExceptionCode();
            }
            RpcEndExcept

            MIDL_user_free(pszFileName);

            Sample_RpcUnBinding();
        }

        return ret;
    }

    //////////// Sample_QssSetOrder() ////////////

    long Sample_QssSetOrder(
        QSS_CLIENT_INFO      ClientInfo,
        QSS_ORDER_PARAM      OrderParam)
    {
        long      ret;
        unsigned long ulCode;

        // Return value
        // RPC error code

        ret = Sample_RpcBinding();

        if (0 == ret)
        {
            // Procedure of exception handling: RpcTryExcept, RpcExcept, RpcEndExcept
            RpcTryExcept
            {
                ret = QssSetOrder(ClientInfo, OrderParam);
            }
            RpcExcept(1)
            {
                // RPC error handling
                ulCode = RpcExceptionCode();
            }
            RpcEndExcept

            Sample_RpcUnBinding();
        }
    }

```

```

        return ret;
    }

    //////////// Sample_QssGetOrderState() ////////////

    long Sample_QssGetOrderState(
        QSS_CLIENT_INFO ClientInfo,
        short            Flag,
        long             BufferNum,
        long __RPC_FAR  *OrderNum,
        QSS_ORDER_STATE __RPC_FAR *OrderState)
    {
        long    ret;                // Return value
        unsigned long ulCode;        // RPC error code
        ret = Sample_RpcBinding();

        if (0 == ret)
        {
            // Procedure of exception handling : RpcTryExcept, RpcExcept, RpcEndExcept
            RpcTryExcept
            {
                ret = QssGetOrderState(ClientInfo, Flag, BufferNum, OrderNum, OrderState);
            }
            RpcExcept(1)
            {
                // RPC error handling
                ulCode = RpcExceptionCode();
            }
            RpcEndExcept

            Sample_RpcUnBinding();
        }

        return ret;
    }

    //////////// Sample_PipeAlloc() ////////////

    void Sample_PipeAlloc(
        char            *pcStateInfo,        // PIPE_STATE structure
        unsigned long    nReqSize,            // Size of the buffer QSS requests
        unsigned char    **pcAllocatedBuffer, // Pointer to the buffer for data transfer
        unsigned long    *pnAllocatedSize)    // Size of the buffer for data transfer
    {
        ////////////
        // Return the pointer to the buffer for data transfer
        ////////////

        if (nReqSize > (CLIENT_BUFFER_SIZE * sizeof(unsigned char)))
        {
            *pnAllocatedSize = CLIENT_BUFFER_SIZE * sizeof(unsigned char);
        }
        else
        {
            *pnAllocatedSize = nReqSize;
        }

        *pcAllocatedBuffer = g_pcBuffer;
    }

    //////////// Sample_PipePull() ////////////

```

```
void Sample_PipePull(
    char *pcStateInfo,           // PIPE_STATE structure
    unsigned char *pcBuffer,     // Pointer to the buffer for data
    unsigned long nMaxBufferSize, // Maximum size of buffer
    unsigned long *pnSizeToSend) // Size of data transferred (Size of the data copied to the
    buffer)
{
    //////////////////////////////////////
    // Transfer the data read from image file to QSS
    //////////////////////////////////////

    PIPE_STATE *psState = (PIPE_STATE*)pcStateInfo;

    if (psState->hFile == NULL)
    {
        // Open the file.
        if(NULL == (psState->hFile = fopen(psState->pszFileName, TEXT("rb"))))
        {
            return;
        }
    }

    // Store the size defined in nMaxBufferSize to pcBuffer.
    *pnSizeToSend = fread(
        pcBuffer,
        sizeof(char),
        nMaxBufferSize,
        psState->hFile);

    if (0 == *pnSizeToSend)
    {
        // It is required to set pnSizeToSend to 0 when the file transfer is over.

        // Reading file is complete.
        fclose(psState->hFile);
        psState->hFile = NULL;
    }
}
```

## Chapter 2. Application Programming Interface Reference

This chapter describes the QSS Network Service API ("QSS API") that is the Application Programming Interface (API) to control the QSS.

You may call the QSS API from the Client (peripheral device or application) that is connected to QSS via Ethernet. QSS API can be called from the Client via RPC (Remote Procedure Call) by a Client. In order to achieve this, Client has to go through the RPC procedure before it calls QSS API. Refer to "1-5. Build Client Application" for RPC and RPC procedure.

Below is the list of API provided for QSS Network Service.

Print reception API's:

Functions	Description
<b>QssGetName</b>	Get QSS model name and interface version.
<b>QssTransmitFile</b>	Send print data to QSS.
<b>QssSetOrder</b>	Spool the order.
<b>QssAbort</b>	Cancel the order spooled.
<b>QssSetPUInfo</b>	Send information to be printed on pricing sheet to QSS
<b>QssGetPaper</b>	Get paper information registered.
<b>QssGetError</b>	Get error or attention message currently occurs or is displayed on QSS.
<b>QssGetOrderState</b>	Get the status of the order spooled.
<b>QssGetPrinterState</b>	Get the current status of QSS.
<b>QssGetChannellInfo</b>	Get the print channel information.
<b>QssGetSumInfo</b>	Get the total number of print and/or total amount of replenisher.
<b>QssGetProfile</b>	Get the QSS profile information.
<b>QssLookup</b>	Search for QSS connected to Network.
<b>QssAbortRefld</b> $\Delta 1$	Cancel the spooled order based on the reference number.
<b>QssGetOrderStateRefld</b> $\Delta 2$	Get the status of spooled order based on the reference number.
<b>QssGetOrderHistory</b> $\Delta 2$	Get order history.
<b>QssTransmitFile2</b> $\Delta 9$	Send print data to QSS.
<b>QssSetOrder2</b> $\Delta 9$	Spool order.

---

## QssGetName

Get QSS model name and I/F version.

```
long QssGetName (
    QSS_PRINTER_INFO      *QssInfo // [Output] QSS information
);
```

### Parameter

QssInfo

Get QSS information (QSS\_PRINTER\_INFO).

### Return value

Return QSS\_SUCCESS when succeeded, else return QSS\_FAIL.

### Description

Used to check the QSS model name and interface version.

---

## QssTransmitFile

Send print data to QSS.

```
long QssTransmitFile(
    QSS_CLIENT_INFO ClientInfo,      // [Input] Client Information
    QSS_FRAME_PIPE Pipe,            // [Input] RPC pipe
    QSS_FRAME_PARAM FrameParam      // [Input] Frame print parameter
);
```

### Parameters

ClientInfo

Set the Client information (QSS\_CLIENT\_INFO) of the caller.

Pipe

Set the RPC pipe (QSS\_FRAME\_PIPE) to send image files.

FrameParam

Set the image print parameter (QSS\_FRAME\_PARAM).

### Return value

Return QSS\_SUCCESS when succeeded, else return either of the following:

QSS_INVALID_ORDERNO:	Invalid Request number
QSS_INVALID_FRAMENO:	Invalid frame number
QSS_NOT_SUPPORT_FORMAT:	Image format not supported
QSS_INVALID_REPEATNUM:	Invalid repeat number
QSS_DISKFULL_SPOOL:	Insufficient free disk space in spool area
QSS_DISABLE_MODE:	Orders are not accepted in the selected mode. Δ10-1
QSS_RECEIVE_ABORT:	Order rejected. (e.g. deleted by QSS while receiving)

**Description**

When Client requests QSS to print, it is required to send print data (both images and parameters needed to make prints) to QSS by calling QssTransmitFile.

The print data sent from Client by calling QssTransmitFile is copied to the spool area of QSS. Printing does not start yet at this stage. It is required to call QssSetOrder to start printing.

The print data stored in the spool area will be deleted when the printing of the image is complete. It will also be deleted when QssSetOrder is not called though 10 minutes has elapsed since the print data was stored in the spool area.

---

**QssSetOrder**

Spool the order.

```
long QssSetOrder(
    QSS_CLIENT_INFO ClientInfo,      // [Input] Client information
    QSS_ORDER_PARAM OrderParam      // [Input] Order Print Parameter
);
```

**Parameters**

ClientInfo

Set the client information (QSS\_CLIENT\_INFO) of the caller.

OrderParam

Set the print parameter of the order (QSS\_ORDER\_PARAM).

**Return value**

Return QSS\_SUCCESS when succeeded, else return either of the following:

QSS_INVALID_ORDERNO:	Invalid Request number
QSS_INVALID_PAPER:	Invalid paper
QSS_INVALID_FRAMENUM:	Invalid number of frames
QSS_INVALID_WBSIZE:	Invalid white border size
QSS_INVALID_INDEXSIZE:	Invalid index print size
QSS_INVALID_PAPERFITTING:	Invalid paper fitting
QSS_INVALID_PAPERLENGTH:	Invalid paper advance length
QSS_RECEIVE_ABORT:	Receipt denied $\Delta 7$ (e.g. deleted by QSS during the receipt process)

**Description**

QSS controls print request by order. Therefore, it is necessary to call QssSetOrder to spool the order after image files are sent with QssTransmitFile. Order will be copied to the spool area via QssSetOrder and wait to be printed.

---

**QssAbort**

Cancel the spooled order.

```

long QssAbort(
    QSS_CLIENT_INFO ClientInfo,    // [Input] Client information
    unsigned short OrderNo        // [Input] Request number of the order to be deleted
);

```

**Parameters**

ClientInfo

Set the Client information (QSS\_CLIENT\_INFO) of the caller.

OrderNo

Specify the Request number of the order to be deleted.

The range is 0 – 65534.

△6

**Return value**

Return QSS\_SUCCESS when succeeded, else return either of the following:

QSS\_INVALID\_ID\_AUTHORITY: No authority to delete

QSS\_NO\_SUCH\_ORDER: Order not found

**Description**

You may delete the spooled order that is waiting to be printed or being printed together with its print data.

In case the order being printed is to be deleted, QSS will implement the print cancel process, and upon completion of this process the order will be deleted. QssAbort will notify the Client of the receipt of the request without waiting for the completion of print cancel process. You may confirm whether the order has actually been deleted or not by calling QssGetOrderState.

**QssSetPUInfo**

Send information to be printed on pricing sheet to QSS

```

long QssSetPUInfo(
    QSS_CLIENT_INFO ClientInfo,    // [Input] Client information
    unsigned short OrderNo,        // [Input] Request number
    QSS_PU_INFO Pinfo             // [Input] PU output information
);

```

**Parameters**

ClientInfo

Set the Client information (QSS\_CLIENT\_INFO) of the caller.

OrderNo

Specify the Request number whose pricing sheet is to be printed out on PU.

The range is 0 – 65534.

△6

Pinfo

Set the PU information to be output by PU (QSS\_PU\_INFO).

**Return value**

Return QSS\_SUCCESS when succeeded, else return the following:

QSS\_NOT\_CONNECTED\_PU: PU not installed.

### Description

Send the information to be used to issue a pricing sheet on PU (Pricing Unit) connected to the QSS. The pricing sheet is printed out upon completion of printing an order. (A pricing sheet per order)

PU is an optional accessory of QSS. When PU is not registered in Option Registration mode of the QSS, or when the specified PU is not of the type that will print out the receipt on the PU, QSS\_NOT\_CONNECTED\_PU is returned as return value. Δ19 -1

You may register PU as an optional accessory in Option Registration mode of QSS.

---

## QssGetPaper

Get paper information registered.

```
long QssGetPaper(
    short          Flag,           // [Input]  Get flag
    long           BufferNum,      // [Input]  Number of buffer for paper information
    long           *PaperNum,     // [Output] Number of paper magazine installed /
                                //           Number of paper magazine registered
    QSS_PAPER_INFO *PaperInfo     // [Output] Paper information
);
```

### Parameters

Flag

Define the information you want to get. You can specify the values to flag as follows:

0: Get the paper information of the paper magazine currently installed

1: Get the paper information of the paper magazine registered

BufferNum

Define the number of buffer of paperInfo

PaperNum

Returns the following information according to the value set for Flag.

When flag is set to 0: Number of paper magazines currently installed

When flag is set to 1: Number of paper magazine registered

PaperInfo

According to the value set in the flag, the following information will be returned. Number of communication to return the information depends on the setting of BufferNum.

When Flag is set to 0: Information of the paper in the paper magazine currently installed (QSS\_PAPER\_INFO)

When Flag is set to 1: Information of the paper registered (QSS\_PAPER\_INFO)

### Return value

Return QSS\_SUCCESS when succeeded, else return QSS\_FAIL.

Return QSS\_REMAINING\_DATA when there is information not acquired yet.

**Description**

With Get flag, the paper information of the paper magazine currently installed on the QSS or that of the ones registered to QSS can be acquired,. Client is required to call QssGetPaper to check the paper currently registered when making the order parameter setting.

When you call this function with 0 set to BufferNum, the number of paper magazine currently installed or registered is returned in PaperNum.

When the number in BufferNum is smaller than the number of the paper currently installed or registered, then "There is information that is not acquired yet." is returned as return value.

---

**QssGetError**

Get error or attention message currently given on QSS.

```
long QssGetError(
    short      Flag,           // [Input]  Get flag
    long      BufferNum,       // [Input]  Buffer number of Error information
    long      *ErrorNum,       // [Output] Number of errors and/or attention
                                messages currently occur
    QSS_ERROR_INFO*ErrorInfo   // [Output] Error information
);
```

**Parameters**

Flag

Define the information you want to get. You can set either the following:

- 0: Get error information only.
- 1: Get attention messages only.
- 2: Get both error and attention messages.

BufferNum

Define the number of buffer for ErrorInfo.

ErrorNum

Return the number of errors and attention messages currently given.

ErrorInfo

Return the errors and attention messages currently given (QSS\_ERROR\_INFO). Number of communication to return the information depends on the setting of BufferNum.

**Return value**

Return QSS\_SUCCESS when succeeded, else return QSS\_FAIL.

Return QSS\_REMAINING\_DATA when there is information not acquired yet.

**Description**

When called with 0 set in BufferNum, number of errors or attention messages currently given on QSS is returned in ErrorNum.

When the value in BufferNum is smaller than the number of errors or attention messages that are

currently given on QSS, then “There is information that is not acquired yet.” is returned as return value.

## QssGetOrderState

Get the status of the order spooled.

```
long QssGetOrderState(
    QSS_CLIENT_INFO    ClientInfo,    // [Input] Client information
    short              Flag,          // [Input] Get flag
    long               BufferNum,      // [Input] Number of buffer for order status
    long               *OrderNum,      // [Output] Number of order acquired
    QSS_ORDER_STATE    *OrderState    // [Input/ Output] Order status
);
```

### Parameters

ClientInfo

Set the Client information (QSS\_CLIENT\_INFO) of the caller.

Flag

Define the information you want to get as follows:

0: You can get the status of the order that you specify among all the orders Client has.

1: You can get the status of all the orders that the Client has and that are stored in the spool.

BufferNum

Define the number of buffer for OrderState.

The range is 0 – 10000.

△4

OrderNum

Depending on the value set in the flag, the following information will be returned.

When 0 is set for Flag: Number of order acquired is returned.

When 1 is set for Flag: Number of order the caller Client currently has is returned.

OrderState

Depending on the value set in the flag, the following information is returned. Number of communication to return the information depends on the setting of BufferNum.

When 0 is set in the flag: Status of the order that is specified in OrderNo of OrderState (QSS\_ORDER\_STATE)

When 1 is set in the flag: Status of the spooled order that Client has. (QSS\_ORDER\_STATE)

### Return value

Return QSS\_SUCCESS when succeeded else return QSS\_FAIL.

Return QSS\_REMAINING\_DATA when there is information not acquired yet.

### Description

Use QssGetOrderState to check the status of the order. Client may know the current status of the order is either the following:

Receiving order

Print queue

Printing

Canceling

Suspended

Printed

△2

Canceled

When called with 0 set in BufferNum, the number of the corresponding orders is returned in OrderNum.

When the value in BufferNum is smaller than the number of corresponding orders, "There is information that is not acquired yet." is returned as return value.

## QssGetPrinterState

Get the current status of QSS.

```
long QssGetPrinterState(
    QSS_PRINTER_STATE    *PrinterState    // [Output] Status of QSS
);
```

### Parameter

PrinterState

Returns the current status of QSS (QSS\_PRINTER\_STATE).

### Return value

Return QSS\_SUCCESS when succeeded else return QSS\_FAIL.

### Description

It is required for the Client to call QssGetPrinterState to check the current status of QSS before requesting the printing operation.

## QssGetChannelInfo

Get the print channel information.

```
long QssGetChannelInfo (
    long    BufferNum,    // [Input]  Number of buffer for print channel information
    long    *ChannelNum, // [Output] Number of print channel information
    QSS_PRINT_CHANNEL *PrintChannel // [Output] Print channel information
);
```

### Parameters

BufferNum

Define the number of buffer in PrintChannel.

ChannelNum

Return the number of print channel information defined.

PrintChannel

Return the print channel information defined (QSS\_PRINT\_CHANNEL). Number of communication to return the information depends on the setting of BufferNum.

**Return value**

Return QSS\_SUCCESS when succeeded else return QSS\_FAIL.

Return QSS\_REMAINING\_DATA when there is information not acquired yet.

**Description**

Use QssGetChannelInfo to check the print channel information.

When called with 0 set in BufferNum, the number of print channel information currently defined is returned in ChannelNum.

If the value in BufferNum is smaller than the number of print channel information, "There is information that is not acquired yet." is returned as return value.

---

**QssGetSumInfo**

Get the total number of prints made on QSS and total amount of replenisher solution

```
long QssGetSumInfo (  
    QSS_SUM_INFO *SumInfo // [Output] Total information  
);
```

**Parameters**

SumInfo

Return the total number of prints made on the QSS and total amount of solution replenished (QSS\_SUM\_INFO).

**Return value**

Return QSS\_SUCCESS when succeeded, else return QSS\_FAIL.

**Description**

QssGetSumInfo is used to get the number of prints made on the QSS and total amount of replenisher solution.

---

**QssGetProfile**

Get the QSS profile information.

```
long QssGetProfile(  
    QSS_PROFILE_INFO* ProfileInfo, // [Input] Get profile information  
    QSS_PROFILE_PIPE Pipe // [Output] RPC pipe  
);
```

**Parameters**

ProfileInfo

Define the profile information you wish to get (QSS\_PROFILE\_INFO).

Pipe

Define the RPC pipe (QSS\_CMS\_PIPE) to be used to get the ICC profile.

**Return value**

Return QSS\_SUCCESS when succeeded, else return QSS\_FAIL.

Return QSS\_NOTEXIST\_PROFILE when the profile defined does not exist.

**Description**

Get the monitor and/or printer profile that is used for Color Management System (CMS) of QSS.

Printer profile exists for every paper width and surface.

The profile you will get is the data of International Color Consortium (ICC) profile format.

**QssLookup**

Search for QSS on which Net Order is available and return the information of the QSS.

**NOTE: This function is provided in QSSOpt.DLL.**

```
long QssLookup(
    unsigned short*      BufferNum,      // [Input] Number of buffer for QssInfo
    unsigned short*      QssNum,        // [Output] Number of QSS that is found
    QSS_PRINTER_ENUM*    QssInfo        // [Output] Get profile information
);
```

**Parameters**

BufferNum

Define the number of buffer for QssInfo.

QssNnum

Return the number of QSS connected via Network.

QssInfo

Return QSS information (QSS\_PRINTER\_ENUM structure).

**Return value**

Return SUCCESS (0) when succeeded, else return FALSE (1).

Return QSS\_REMAINING\_DATA (24) when there is not enough buffer for QssInfo.

**Description**

Call the function with 0 set in BufferNum first, and get the number of QSS connected. Then prepare the buffer for each QSS from which QssInfo is obtained and call the function to get the QSS information. When QSS\_REMAINING\_DATA is returned, call the function again. When SUCCESS is returned, it means you have got the information of all the QSS connected via Network.

**QssAbortRefId** △1

Cancel the spooled order based on the reference number.

```
long QssAbortRefId(
    QSS_CLIENT_INFO      ClientInfo,    // [Input] Client information
```

```

        unsigned hyper      RefId      // [Input] Reference number of order to be deleted
    );

```

### Parameters

ClientInfo

Set the Client information (QSS\_CLIENT\_INFO) of the caller.

RefId

Set the reference number of the order to delete.

The range is 1 – 999999999999999999 (19 digits).

△6

### Return value

Return QSS\_SUCCESS when succeeded, else return either of the following:

QSS_INVALID_ORDERNO:	Invalid Request number
QSS_INVALID_FRAMENO:	Invalid frame number
QSS_NOT_SUPPORT_FORMAT:	Image format not supported
QSS_INVALID_REPEATNUM:	Invalid repeat number
QSS_DISKFULL_SPOOL:	Insufficient free disk space in spool area.
QSS_DISABLE_MODE:	Not Net Order mode.
QSS_RECEIVE_ABORT:	Order rejected.
	(e.g. deleted by QSS while receiving)

### Description

When Client requests QSS to print, it is required to send print data (both images and parameters needed to make prints) to QSS by calling QssTransmitFile.

The print data sent from Client by calling QssTransmitFile is copied to the spool area of QSS. Printing does not start yet at this stage. It is required to call QssSetOrder to start printing.

The print data stored in the spool area will be deleted when the printing of the image is complete. It will also be deleted when QssSetOrder is not called though 10 minutes has elapsed since the print data was stored in the spool area.

---

## QssGetOrderStateRefId △2

Get the status of spooled order based on the reference number.

```

long QssGetOrderStateRefId(
    QSS_CLIENT_INFO      ClientInfo,    // [Input] Client information
    short                Flag,          // [Input] Get flag
    long                 BufferNum,      // [Input] Number of buffer for order status
    long                 *OrderNum,     // [Output] Number of order whose status
                                         information has been got
    QSS_ORDER_STATE_EX *OrderState      // [Input/Output] Extended order status  △3
);

```

### Parameters

ClientInfo

Set the Client information (QSS\_CLIENT\_INFO) of the caller Client.

**Flag**

Define the information you want to get as follows:

0: You can get the status of the order that you specify among all the orders Client has.

1: You can get the status of all the orders that the Client has and that are stored in the spool.

**BufferNum**

Define the number of buffer for OrderState.

The range is 0 – 10000.

△4

**OrderNum**

Depending on the value set in the flag, the following information will be returned.

When 0 is set for Flag: Number of order acquired is returned.

When 1 is set for Flag: Number of order the caller Client currently has is returned.

**OrderStateEx**

△3

Depending on the value set in the flag, the following information is returned. Number of communication to return the information depends on the setting of BufferNum.

When 0 is set in the flag: Status of the extended order that is specified in RefId of OrderState (QSS\_ORDER\_STATE\_EX)

When 1 is set in the flag: Status of the extended order in the spool that Client has. (QSS\_ORDER\_STATE\_EX)

**Return value**

Return QSS\_SUCCESS when succeeded else return QSS\_FAIL.

Return QSS\_REMAINING\_DATA when there is information not acquired yet.

**Description**

Use QssGetOrderStateRefId to check the status of the order. Client may know the current status of the order is either the following:

- Receiving order
- Print queue
- Printing
- Canceling
- Suspended
- Printed
- Canceled

When called with 0 set in BufferNum, the number of the corresponding orders is returned in OrderNum.

When the value in BufferNum is smaller than the number of corresponding orders, "There is information that is not acquired yet." is returned as return value.

---

**QssGetOrderHistory** △2

Get order history.

```
long QssGetOrderHistory(
    QSS_CLIENT_INFO    ClientInfo,    // [Input] Client information
```

```

        QSS_DATETIME      Datetime,      // [Input] Receipt date
        short             OrderStatus,    // [Input] Order result
        long              BufferNum,      // [Input] Number buffer for order history
        long              *OrderHistoryNum, // [Output] Number of order acquired
        QSS_ORDER_HISTORY *OrderHistory   // [Output] Order history
    );

```

### Parameters

ClientInfo

Set the Client information (QSS\_CLIENT\_INFO) of the caller.

Datetime

Define the day when QSS has received the order (QSS\_DATETIME) as a condition to get order history. Year, month, and day must be defined. This is mandatory.

OrderStatus

Define the type of order – either printed or canceled order - you wish to get history of as a condition to get order history as follows. When 0 is defined, order history returned will include both printed and canceled orders.

Value	Description
QSS_ORDER_STATUS_PRINTED	Printed order
QSS_ORDER_STATUS_CANCELED	Canceled order

BufferNum

Define the number of buffer for OrderHistory.

OrderHistoryNum

Number of orders that meet the conditions specified in Datetime and OrderStatus.

OrderHistory

Order history that meets the conditions specified is returned. Number of communication to return the information depends on the setting of BufferNum.

### Return value

Return QSS\_SUCCESS when succeeded else return QSS\_FAIL.

Return QSS\_REMAINING\_DATA when there is information not acquired yet.

### Description

Use QssGetOrderHistory to get order history. Client may know the history of orders that are in QSS\_ORDER\_HISTORY structure.

When called with 0 set in BufferNum, the number of the corresponding orders is returned in OrderHistoryNum.

When the value in BufferNum is smaller than the number of corresponding orders, "There is information that is not acquired yet." is returned as return value.

---

## QssTransmitFile2 △9

Send print data to QSS. By the combination of QssTransmitFile2 and QssSetOrder2 Fast Print function will

be made available.  $\Delta$ 12-2

```
long QssTransmitFile2(
    QSS_CLIENT_INFO      ClientInfo,    // [Input] Client information
    QSS_FRAME_PIPE       Pipe,          // [Input] RPC pipe
    QSS_FRAME_PARAM2     FrameParam2    // [Input] Frame print parameter
);
```

### Parameters

ClientInfo

Set the Client information (QSS\_CLIENT\_INFO) of the caller Client.

Pipe

Set the RPC pipe (QSS\_FRAME\_PIPE) to send image files.

FrameParam2

Set print parameter of the image (QSS\_FRAME\_PARAM2).

### Return value

Return QSS\_SUCCESS when succeeded, else return either of the following:

QSS_INVALID_ORDERNO:	Invalid Request number
QSS_INVALID_FRAMENO:	Invalid frame number
QSS_NOT_SUPPORT_FORMAT:	Image format not supported
QSS_INVALID_REPEATNUM:	Invalid repeat number
QSS_DISKFULL_SPOOL:	Insufficient free disk space in spool area
QSS_RECEIVE_ABORT:	Order rejected (e.g. deleted by QSS while receiving)
QSS_ILLEGAL_IMAGEDATA:	Illegal image file $\Delta$ 10-2
QSS_INVALID_IMAGESIZE:	Illegal image file size $\Delta$ 10-3
QSS_INVALID_PARAMETER:	Invalid parameter $\Delta$ 20-1

### Description

QSS will start printing upon its receipt of print data as far as it is ready to print. For that purpose, it is required to call QssSetOrder2 to spool the order before sending print data (that consists of image to print and parameter required for printing) to QSS by calling QssTransmitFile2.

This is not applicable to QSS-28, QSS-29, and QSS-30.

## QssSetOrder2 $\Delta$ 9

Spool order. By the combination of QssSetOrder2 and QssTransmitFile2 Fast Print function will be made available.  $\Delta$ 12-3

```
long QssSetOrder2(
    QSS_CLIENT_INFO      ClientInfo,    // [Input] Client information
    QSS_ORDER_PARAM2     OrderParam2    // [Input] Order print parameter
);
```

### Parameters

ClientInfo

Set the Client information (QSS\_CLIENT\_INFO) of the caller Client.

OrderParam2

Set print parameter of the order (QSS\_ORDER\_PARAM2).

### Return value

Return QSS\_SUCCESS when succeeded, else return either of the following:

QSS_INVALID_ORDERNO:	Invalid Request number
QSS_INVALID_PAPER:	Invalid paper
QSS_INVALID_FRAMENUM:	Invalid number of frames
QSS_INVALID_WBSIZE:	Invalid white border size
QSS_INVALID_INDEXSIZE:	Invalid index print size
QSS_INVALID_PAPERFITTING:	Invalid paper fitting
QSS_INVALID_PAPERLENGTH:	Invalid paper advance length
QSS_DISABLE_MODE:	Orders are not accepted in the selected mode. Δ10-4
QSS_RECEIVE_ABORT:	Order rejected (e.g. deleted by QSS while receiving)
QSS_INVALID_OUTMEDIA_PARAM:	Invalid parameter for media output Δ18-1

### Description

QSS manages print request on an order basis, so it is required to call QssSetOrder2 to spool orders.

QSS will start printing upon its receipt of print data as far as it is ready to print. For that purpose, it is required to first call QssSetOrder2 to spool the order then to call QssTransmitFile2 to copy the print data (that consists of print image and parameter required for printing) to the spool area of QSS.

This is not applicable to QSS-28, QSS-29, and QSS-30.

---

## QSS\_PRINTER\_INFO structure

```
typedef struct _QSS_PRINTER_INFO {
    char                Name[20];
    unsigned long       Version;
    unsigned char       IPAddress[4];           // Version 1.0.5      Δ3
    unsigned short      SystemInfo;            Δ18-2
    unsigned char       Reserve[34];
} QSS_PRINTER_INFO;
```

### Member

Name	[Output]
Set the model name of QSS. The string is NULL terminated.	
e.g.) QSS-2801 -> "QSS-28" QSS-2901 -> "QSS-29"	
Version	[Output]
The current version of QSS Network Service is set in hexadecimal.	
e.g.) If the version of QSS Network Service is 1.2.3, 0x01020300 is set.	
IPAddress	[Output] Δ3

Set the IP address of QSS.

SystemInfo

[Output] △18-3

Define the running status of the QSS as follows:

Value	Description
QSS_SYSTEM_INFO_QSS	Running as QSS
QSS_SYSTEM_INFO_DDP	Running as dDP

Reserve

Reserved (Unused)

## QSS\_CLIENT\_INFO structure

```
typedef struct _QSS_CLIENT_INFO {
    char                User[20];
    char                Host[20];
    unsigned char       Address[6];
    unsigned char       IPAddress[4];           // Version 1.0.5      △2
    unsigned short      Port;                  // Version 1.0.5
    unsigned long        Version;              // Version 1.0.5
    unsigned short      Level;                 // Version 1.0.5
    unsigned char       Reserve[38];
} QSS_CLIENT_INFO;
```

### Member

User [Input]

Define the user name.  
This string should be a maximum of 19 characters and NULL terminated.

Host [Input]

Define the host name.  
This string should be a maximum of 19 characters and NULL terminated.

Address [Input]

Define MAC address.

IPAddress [Input] △2

Define IP address of Client host PC.

Port [Input] △2

Define the port number of the socket to receive event notification.

Version [Input] △2

Define the version of NetOrder API to use.

Level [Input] △2

Define client level.

Value	Description
QSS_CLIENT_LEVEL1	Status of orders that the Client has sent to QSS is received in order status notification from QSS.
QSS_CLIENT_LEVEL2	Status of all orders is received in order status notification from QSS.

Reserve

Reserved (Unused)

### Remarks

Use this structure as the information when QSS manages and controls the orders.  
Purpose of this structure is to serve as an authentication at the time of canceling an order, and as identification at the time of confirming an order from QSS order management display.

## QSS\_FRAME\_PIPE structure

```
typedef struct pipe_QSS_FRAME_PIPE
```

```

{
    void    (__RPC_FAR * pull) (
        char __RPC_FAR * state,
        unsigned char __RPC_FAR * buf,
        unsigned long esize,
        unsigned long __RPC_FAR * ecount
    );
    void    (__RPC_FAR * push) (
        char __RPC_FAR * state,
        unsigned char __RPC_FAR * buf,
        unsigned long ecount
    );
    void    (__RPC_FAR * alloc) (
        char __RPC_FAR * state,
        unsigned long bsize,
        unsigned char __RPC_FAR * __RPC_FAR * buf,
        unsigned long __RPC_FAR * bcount
    );
    char __RPC_FAR * state;
} QSS_FRAME_PIPE;

```

**Member**

Pull	[Input]
Set the pointer to the image data transfer function implemented by Client. For image data transfer function, implement the logic to transfer the image data to the memory on QSS. The actual image data transfer is implemented by RPC pipe.	
Push	[Input]
Unused. Set NULL.	
Alloc	[Input]
Set the pointer to the get memory size function implemented by Client. For the get memory size function, implement the logic to define the memory size to secure the memory area in the image data destination.	
State	[Input]
Set the Client specific data. Refer to "Microsoft Platform SDK Documentation" for the details of RPC pipe.	

**QSS\_FRAME\_PARAM structure**

```

typedef struct _QSS_FRAME_PARAM {
    unsigned short    OrderNo;
    unsigned short    FrameNum;
    unsigned short    FrameNo;
    char              FileName[18];
    unsigned long      FileSize;
    unsigned long      ImageFormat;
    unsigned short     PrintSize;
    unsigned short     RepeatNum;
    unsigned short     RepeatPos;
    char               CvpString1[120];
    char               CvpString2[120];
    unsigned short     CvpFlg;
}

```

unsigned short	PaperWidth;	// Version 1.0.4	△1
unsigned short	PaperLength;	// Version 1.0.5	△3
unsigned short	Surface;	// Version 1.0.4	△1
unsigned short	WithBorder;		△23-1
unsigned short	PaperFittingFlg;		△23-2
unsigned short	ImageXPixels;	// (Unused)	
unsigned short	ImageYPixels;	// (Unused)	
unsigned short	Reserve1;	// (Unused)	
unsigned hyper	RefId;	// Version 1.0. 4	△1
unsigned short	SizeRate;	// (Unused)	
unsigned short	Rotate;	// (Unused)	
short	CenterX;	// (Unused)	
short	CenterY;	// (Unused)	
unsigned char	Way;	// (Unused)	△23-3
unsigned char	Reserve2;	// (Unused)	△23-4
unsigned short	EnablePaperFittingFlg;		△23-5
unsigned char	Reserve[4];	// (Unused)	

} QSS\_FRAME\_PARAM;

### Member

OrderNo	[Input]
Request number	
The range is 0 – 65535. When 65535 (0xFFFF) is defined, an order will be added using the reference number as the administration key.	
	△6
FrameNum	[Input]
Define the total number of frames that make up an order.	
The range is 1 – 999.	
FrameNo	[Input]
Define the frame number.	
The range is 1 – 999.	
FileName	[Input]
Define the file name of the image to be sent to QSS. (Mainly used for index print)	
This string should be a maximum of 17 characters and NULL terminated.	
FileSize	[Input]
Define the file size of the image to be sent to QSS (unit: Byte).	
ImageFormat	[Input]
Define the format of the image to be sent to QSS.	
Define one of the formats defined in SupportImageFormat in QSS_PRINTER_STATE structure by calling QssGetPrinterState function.	
Any image format can be defined for each individual frame.	
PrintSize	[Input]
Define the print size as follows:	
	△3
Value	Description
QSS_PRINT_SIZE_C	Values of PaperWidth, PaperLengthC, Surface, and WithBorderC of QSS_ORDER_PARAM structure are adopted.
QSS_PRINT_SIZE_P	Values of PaperWidth, PaperLengthP, Surface, and WithBorderP of QSS_ORDER_PARAM structure are adopted.
QSS_PRINT_SIZE_H	Values of PaperWidth, PaperLengthH, Surface, and WithBorderH of QSS_ORDER_PARAM structure are adopted.
QSS_PRINT_SIZE_FREE_C	Values of PaperWidth, PaperLength, and Surface of this structure and value of WithBorderC of QSS_ORDER_PARAM structure are adopted.
QSS_PRINT_SIZE_FREE_P	Values of PaperWidth, PaperLength, and Surface of this structure and value of WithBorderP of QSS_ORDER_PARAM structure are adopted.
QSS_PRINT_SIZE_FREE_H	Values of PaperWidth, PaperLength, and Surface of this structure and value of WithBorderH of QSS_ORDER_PARAM structure are adopted.
RepeatNum	[Input]

Define the number of repeat prints to be made.

The range is 0 – 999.

NOTE: When you define 0, the frame will not be printed, but will be printed on the index print.

Allowable range is 1 – 9999 with NetOrder API version 2.3.0 and up.

△26-1

RepeatPos [Input]

Define where in the CVP printing to include the repeat count (sequential serial number).

0 - 117: The 1<sup>st</sup> line of CVP

120 – 237: The 2<sup>nd</sup> line of CVP printing

255: No repeat count included in CVP printing

CvpString1 [Input]

CvpString2 [Input]

Define the string to be printed as CVP.

CvpString1: String to be printed on the 1<sup>st</sup> line of CVP printing

CvpString2: String to be printed on the 2<sup>nd</sup> line of CVP printing

Set arbitrary Noritsu Character Code. The string must be NULL terminated.

You may define up to 115 characters, but the number of characters to be printed as CVP printing depends on the feed length of the paper or the QSS model.

When it is defined by using RepeatPos so the repeat count will be included in the CVP printing, the value of repeat count will supersede the information that is supposed to be printed to the predetermined position (3 digits).

△12-4

CvpFlg [Input]

Determine whether the CVP printing uses QSS data or the value defined in CvpString1 and CvpString2.

Value	Description
QSS_CVP_AUX	CvpString1 and CvpString2 are used for both the 1 <sup>st</sup> and the 2 <sup>nd</sup> lines of CVP.
QSS_CVP_1QSS2AUX	QSS data is used for the 1 <sup>st</sup> line, and CvpString2 is used for the 2 <sup>nd</sup> line.
QSS_CVP_1AUX2QSS	CvpString1 is used for the 1 <sup>st</sup> line, and QSS data is used for the 2 <sup>nd</sup> line.
QSS_CVP_QSS	QSS data is used for both the 1 <sup>st</sup> and the 2 <sup>nd</sup> lines.

PaperWidth [Input] △1

Define the paper width to be printed. (unit: 1/10 mm)

When converting the paper width/length from inch to 1/10mm, refer to “inch – 1/10mm Conversion Table” enclosed later in this document.

You may define the same number of paper widths as that can be installed to the connected QSS.

In case of QSS-31, you may define 2 different paper widths for an order.

In order to use this parameter, be sure to set QSS\_PRINT\_SIZE\_FREE\_C, QSS\_PRINT\_SIZE\_FREE\_P, or QSS\_PRINT\_SIZE\_FREE\_H in PrintSize. △3

PaperLength [Input] △1 △3

Define the paper advance length for each frame (unit: 1/10 mm).

When converting the paper width/length from inch to 1/10mm, please refer to “inch – 1/10mm Conversion Table” enclosed in this document.

In order to use this parameter, be sure to set QSS\_PRINT\_SIZE\_FREE\_C, QSS\_PRINT\_SIZE\_FREE\_P, or QSS\_PRINT\_SIZE\_FREE\_H in PrintSize. △3

Surface [Input] △1

Define the surface type of the paper to use. The range is 1-4.

In order to use this parameter, be sure to set QSS\_PRINT\_SIZE\_FREE\_C, QSS\_PRINT\_SIZE\_FREE\_P, or QSS\_PRINT\_SIZE\_FREE\_H in PrintSize. △3

WithBorder [Input] △23-6

Define the size of the white border that appears on the resulting print. (Range: 0-99, Unit: 1/10 mm)

PaperFittingFlg [Input] △23-7

Specify the paper fitting type. For the detail, refer to “PaperFittingGlg” in the QSS\_ORDER\_PARAM structure.

ImageXPixels

Unused

ImageYPixels

Unused

Reserve1

	Unused	
RefId	[Input]	
	Reference number. You may define any 64-bit identifier. Setting 0xFFFF to Request number (OrderNo) enables to add an order based on the reference number. This also enables to use 64-bit data as administration key in case Client manages orders. You may input any number from 1 to 999999999999999999 (19 digits).	
SizeRate	Unused	
Rotate	Unused	
CenterX	Unused	
CenterY	Unused	
Way		△23-8
Reserve2	Unused	△23-9
EnablePaperFittingFlg		△23-10
	To specify paper fitting type to each individual frame (PaperFittingFlg), set "0"; otherwise set "1".	
Reserve	Reserved (Unused)	

## QSS\_ORDER\_PARAM structure

```
typedef struct _QSS_ORDER_PARAM {
    unsigned short    OrderNo;
    unsigned short    FrameNum;
    unsigned short    PaperWidth;
    unsigned short    PaperLengthC;
    unsigned short    PaperLengthP;
    unsigned short    PaperLengthH;
    unsigned short    Surface;
    unsigned short    WithBorderC;
    unsigned short    WithBorderP;
    unsigned short    WithBorderH;
    unsigned short    IndexPrintFlg;
    unsigned short    PaperFittingFlg;
    unsigned short    IndexPaperWidth;
    unsigned short    IndexSurface;
    unsigned short    CmsFlg;
    unsigned short    Reserve1;           // (Unused)
    unsigned hyper    RefId;             // Version 1.0.4      △1
    unsigned short    SorterNum;         // Version 1.0.6      △5
    unsigned char     Reserve[24];
} QSS_ORDER_PARAM;
```

### Member

OrderNo	[Input]	
	Define the Request number. The range is 0 – 65535. When 65535 (0xFFFF) is defined, an order will be added using the reference number as the administration key.	
FrameNum	[Input]	△6
	Define the total number of frames that make up an order. The range is 1 – 999.	
PaperWidth	[Input]	
	Define the paper width to be printed. (unit: 1/10 mm)	

- PaperLengthC [Input]  
 PaperLengthP [Input]  
 PaperLengthH [Input]  
 Define the paper advance length. (unit: 1/10 mm)  
 PaperLengthC: Paper advance length for Classical size.  
 PaperLengthP: Paper advance length for Panoramic size.  
 PaperLengthH: Paper advance length for High-definition size.  
 NOTE: Get the paper information (QSS\_PAPER\_INFO) with QssGetPaper function, and define the paper feed length between the minimum and maximum feed lengths (PaperLengthMin and PaperLengthMax).
- Surface [Input]  
 Define the surface type of the paper to be printed  
 The range is 1-4 on QSS-28, 29, and 30 series.
- WithBorderC [Input]  
 WithBorderP [Input]  
 WithBorderH [Input]  
 Define the width of the white border on the resultant print. (unit: 1/10 mm)  
 The range is 0 – 99.  
 When 0 is set, the resultant print will be borderless (no white border).  
 WithBorderC: Define the width of the white border for Classical size print.  
 WithBorderP: Define the width of the white border for Panoramic size print.  
 WithBorderH: Define the width of the white border for High-definition size print.
- IndexPrintFlg [Input]  
 Define the paper size of the index print as follows:1

△11-3

Value	Index Size	QSS model					
		28xx	29xx 30xx 31xx (except 3102-2)	3102-2	32xx 34xx	3300	33
QSS_INDEX_NONE	No index print	○	○	○	○	○	○
QSS_INDEX_3HS	3HS (82.5mm x 158mm)	○	○	○	○	○	○
QSS_INDEX_3R	3R (89mm x 127mm)	○	○	○	○	○	○
QSS_INDEX_3HD	3HD (89mm x 158mm)	○	○	○	○	○	○
QSS_INDEX_3W	3W (89mm x 178mm)	×	×	×	×	×	×
QSS_INDEX_3WS	3WS (89mm x 178mm)	○	○	○	○	○	○
QSS_INDEX_4R	4R (102mm x 152mm)	○	○	○	○	○	○
QSS_INDEX_4HD	4HD (102mm x 178mm)	○	○	○	○	○	○
QSS_INDEX_5R	5R (127mm x 178mm)	○	○	○	○	○	○
QSS_INDEX_6R	6R (152mm x 203mm)	○	○	○	○	○	○
QSS_INDEX_6HD	6HD (152mm x 254mm)	○	○	○	○	○	○
QSS_INDEX_6W	6W (152mm x 305mm)	○	○	○	○	○	○
QSS_INDEX_8RS	8RS (203mm x 254mm)	×	○	○	○	○	○
QSS_INDEX_8R	8R (203mm x 305mm)	×	○	○	○	○	○
QSS_INDEX_8HD	8HD (203mm x 356mm)	×	○	○	○	○	○
QSS_INDEX_CD40	CD_40 (120mm x 120mm)	×	○	○	○	○	○
QSS_INDEX_CD40A	CD_40A (89mm x 120mm)	×	○	○	○	○	○
QSS_INDEX_CD40B	CD_40B (102mm x 120mm)	×	○	○	○	○	○
QSS_INDEX_3WL	3WL (89mm x 254mm)	×	○	○	○	○	○
QSS_INDEX_3WL_18	3WL_18 (89mm x 254mm)	×	○	○	○	○	○
QSS_INDEX_4WL_18	4WL_18 (102mm x 254mm)	×	○	○	○	○	○
QSS_INDEX_12R	12R (305mm x 457mm)	×	×	○	○	×	×
△13-1							

Number of frames to be printed on an index print is calculated automatically based on the index print size and number of frames in an order.

PaperFittingFlg [Input]

Define the type of paper fitting to apply to the image, based on the print size.

Value	Description
QSS_PF_CUT	Cut
QSS_PF_WHOLE	Overall
QSS_PF_SAME	Real size

IndexPaperWidth [Input]

Define the width of index print. (unit: 1/10mm)

IndexSurface [Input]

Define the paper surface of the index print.

The range is 1-4 for QSS-28, 29, and 30 series.

CmsFlg [Input]

Define whether CMS of QSS will be applied to the order accepted.

Value	Description
QSS_CMS_ON	CMS conversion will be performed on QSS.
QSS_CMS_OFF	CMS conversion will not be performed on QSS.

Reserve1 Δ1

Unused.

RefId [Input] Δ1

Reference number.

You may define any 64-bit identifier. Setting 0xFFFF to Request number (Order No) enables to add an order based on the reference number.

This also enables to use 64-bit data as administration key in case client manages orders. You may input 1 – 9999999999999999 (19 digits).

SorterNum [Input] Δ5

You may define how many prints are placed on a receiver of the sorter before sorter moves.

The range is 0 – 120.

When you define 0, the sorter will move when the maximum number of prints that a receiver can hold is placed on a receiver.

SorterNum is only available with the NetOrder API of version 1.0.6 and on.

With the earlier version of NetOrder API, this value is fixed to 0.

Client is requested to set the version of the NetOrder API that it uses to Version of QSS\_CLIENT\_INFO.

When the version in use is 1.0.6, set 0x01000600 to Version.

QSS-30 does not support SorterNum. Δ7

Reserve

Reserved (Unused)

Remarks:

For PaperWidth, Surface, and IndexPaperWidth, define that of the registered paper.

You may get the information on registered paper by calling QssGetPaper.

Note:

On the single-magazine type QSS, types of paper for regular print (PaperWidth and Surface) and index print (IndexPaperWidth and IndexSurface) must be identical.

## QSS\_PAPER\_INFO structure

```
typedef struct _QSS_PAPER_INFO {
    unsigned short    PaperWidth;
    unsigned short    Resolut;
    unsigned short    MagazineState;
    unsigned long     PaperRemaind;
    unsigned short    Surface;
    unsigned short    PaperLengthMin;
    unsigned short    PaperLengthMax;
}
```

```
    unsigned char        Reserve[48];
} QSS_PAPER_INFO;
```

**Member**

PaperWidth	[Output]
Define the paper width (unit: 1/10 mm)	
Resolut	[Output]
Define the resolution when exposing the paper. (unit: 1/10 dpi)	
MagazineState	[Output]
Define the presence of paper magazines.	
Value	Description
QSS_MAGAZINE_NONE	No paper magazine installed
QSS_MAGAZINE_A	Magazine A is installed.
QSS_MAGAZINE_B	Magazine B is installed.
QSS_MAGAZINE_C Δ7	Magazine C is installed.
QSS_MAGAZINE_A2 Δ24-1	Magazine A2 is installed.
PaperRemaind	[Output]
Define the length of the remaining paper in the paper magazine (unit: 1/10 mm)	
Valid only when "QSS_MAGAZINE_A" or "QSS_MAGAZINE_B" is set for "MagazineState" else set to "0".	
Valid only when MagazineState is NOT set to QSS_MAGAZINE_NONE.	
In case of QSS_MAGAZINE_NONE, "0" is set.	Δ24-2
Surface	[Output]
Define the paper surface.	
The range is 1-4 for QSS-28, 29, and 30 series.	
PaperLengthMin	[Output]
PaperLengthMax	[Output]
Define the range of the paper feed length. (unit: 1/10 mm)	
PaperLengthMin: Minimum feed length	
PaperLengthMax: Maximum feed length	
Reserve	
Reserved (Unused)	

**QSS\_ERROR\_INFO structure**

```
typedef struct _QSS_ERROR_INFO {
    unsigned short    MainNo;
    unsigned short    SubNo;
    unsigned short    Level;
    wchar_t           Message[256];
    unsigned char     Reserve[26];
} QSS_ERROR_INFO;
```

**Member**

MainNo	[Output]
Define the number of errors and/or attention messages that occur on QSS.	
The range is 1 to 9999. Values 1 to 4999 are for attention messages, while values 5000 to 9999 are for error messages.	
SubNo	[Output]
Define the suffix of error number.	
Level	[Output]
Define the error level as follows:	
Value	Description
QSS_ERROR_LVL1	Error that can be corrected easily
QSS_ERROR_LVL2	Error that needs investigation of the cause and sometimes even needs to call service personnel, such as temperature related error
QSS_ERROR_LVL3	Error that needs to call service personnel, such as PCB malfunction

Message	[Output]
Define the contents of error messages.	
Language depends on the language set for QSS.	
Reserve	
Reserved (Unused)	

## QSS\_ORDER\_STATE structure

```
typedef struct _QSS_ORDER_STATE {
    unsigned short    OrderNo;
    unsigned short    OrderState;
    unsigned short    Reserve1[2]; // (Unused)           Δ1, Δ3
    unsigned hyper    RefId;       // Version 1.0.4      Δ1, Δ3
    QSS_DATETIME      FinishTime;  Δ17-1
    unsigned char     Reserve[18]; Δ1, Δ3
} QSS_ORDER_STATE;
```

### Member

OrderNo	[Input], [Output]
Request number	
When 0 is set for the QssGetOrderState function flag, define the Request number that you wish to get its current status.	
The range is 0 – 65534.	Δ6
When 1 is set for the QssGetOrderState function flag, the Request number that the Client will handle is defined.	
OrderState	[Output]
Define the status of the order as follows:	
Value	Description
QSS_ORDER_ACCEPT	Receiving order
QSS_ORDER_WAIT	Print queue
QSS_ORDER_PRINT	Printing
QSS_ORDER_CANCEL	Canceling
QSS_ORDER_RESERVE	Order reserved
QSS_ORDER_PRINTED	Printed Δ3
QSS_ORDER_CANCELED	Canceled Δ3
QSS_ORDER_NONE	Order does not exist Δ3
Reserve1	Δ1 Δ3
Unused	
RefId	Δ1 Δ3
Reference number	
FinishTime	[Output] Δ17-2
Define the estimated finish time of the order.	
Reserve	
Reserved (Unused)	

## QSS\_ORDER\_STATE\_EX structure Δ3

```
typedef struct _QSS_ORDER_STATE_EX {
    unsigned short    OrderNo;
    unsigned short    OrderState;
    unsigned short    Reserve1[2];
    unsigned hyper    RefId;
    QSS_DATETIME      FinishTime; Δ17-3
    unsigned char     Reserve[6];
} QSS_ORDER_STATE_EX;
```

**Member**

OrderNo	[Input] [Output]
Request number	
QssState	[Output]
Define the status of QSS as follows:	
Value	Description
QSS_ORDER_ACCEPT	Receiving order
QSS_ORDER_WAIT	Print queue
QSS_ORDER_PRINT	Printing
QSS_ORDER_CANCEL	Canceling
QSS_ORDER_RESERVE	Order reserved
QSS_ORDER_PRINTED	Printed
QSS_ORDER_CANCELED	Canceled
QSS_ORDER_NONE	Order does not exist
Reserve1	
Unused.	
Refld	[Input][Output]
Reference number	
When 0 is set to the Flag of QssGetOrderStateRefld function, define the reference number you wish to get the status of.	
The range is 1 – 999999999999999999 (19 digits).	△6
When 1 is set, the reference number that the Client will process is defined.	
FinishTime	[Output] △17-4
Define the estimated finish time of the order.	
Reserve	
Reserved (Unused)	

**QSS\_PRINTER\_STATE structure**

```
typedef struct _QSS_PRINTER_STATE {
```

unsigned short	QssState;	
unsigned short	AbleReceive;	
unsigned short	AblePU;	
QSS_PAPER_INFO	MagazineA;	
QSS_PAPER_INFO	MagazineB;	
unsigned long	SupportImageFormat;	
unsigned hyper	TotalPrintNum;	
unsigned short	TemperatureCD;	
unsigned short	TemperatureBF;	
unsigned short	TemperatureSTB;	
unsigned short	RemaindQuantityCD;	
unsigned short	RemaindQuantityBF;	
unsigned short	RemaindQuantitySTB;	
unsigned hyper	SpoolerSpace;	
unsigned short	IsNetOrderMode;	△13-3
unsigned short	IsCalibrationMode;	△13-4
unsigned short	EnableOutMediaViewer;	△18-4

```
unsigned char Reserve[20];
} QSS_PRINTER_STATE;
```

**Member**

QssState	[Output]
Define the status of QSS as follows:	
Value	Description
QSS_STATE_PRINT	Printing
QSS_STATE_SETUP	Being adjusted (adjusting the solution temperature, Maintenance display being shown, etc.)
QSS_STATE_IDLE	Idling
QSS_STATE_ALERT	Error/Attention message is given

AbleReceive [Output]

Determine whether the input from the external source is printable or not on QSS.

Value	Description
QSS_RECEIVE_ENABLE	Printing of input from external source enabled
QSS_RECEIVE_DISABLE	Printing of input from external source disabled

AblePU [Output]

Determine whether PU connected to QSS is enabled or not.

Value	Description
QSS_PU_ENABLE	PU enabled
QSS_PU_DISABLE	PU disabled

MagazineA [Output]

MagazineB [Output]

Define the information related to the paper magazine installed on QSS.

MagazineA: Magazine A

MagazineB: Magazine B

SupportImageFormat [Output]

Define the image format that QSS supports.

Bit assignment of image format is as follows (Bit 1: Support, Bit 2: Not support):

(There are cases where multiple formats are selected.)

0: JPEG	8: Filmstrip	16: Photo CD	24: Unused
1: BMP	9: FlashPix	17: Photoshop Doc	25: Unused
2: RGB raw	10: PCX	18: Unused	26: Unused
3: RGB raw (16Bit)	11: PICT	19: Unused	27: Unused
4: GIF	12: Pixar	20: Unused	28: Unused
5: TIFF	13: PNG	21: Unused	29: Unused
6: Amiga IFF	14: ScitexCT	22: Unused	30: Unused
7: EPS	15: Targa	23: Unused	31: Unused

e.g.) When QSS supports both JPEG and BMP, the bit assignment will be as follows, and the variable is "3" in decadal system.

31	30	29	28	...	5	4	3	2	1	0	Bit
0	0	0	0		0	0	0	0	1	1	

TotalPrintNum [Output]

Define the total number of prints of the order currently being printed or printed last.

Number of index print is not included.

TemperatureCD [Output]

TemperatureBF [Output]

TemperatureSTB [Output]

Define the current temperature of each processing solution (unit: 0.01 deg C)

TemperatureCD: Define the temperature of CD

TemperatureBF: Define the temperature of BF

TemperatureSTB: Define the temperature of STB

RemaindQuantityCD [Output]

RemaindQuantityBF [Output]

RemaindQuantitySTB [Output]

(Unused)

SpoolerSpace [Output]

Define the free space for the spool. (unit: Byte)

IsNetOrderMode [Output] △13-3

A flag is set to determine whether QSS is currently in the NetOrder mode or not.

Value	Description
QSS_NETORDER_ON	QSS is in NetOrder mode.
QSS_NETORDER_OFF	QSS is not in NetOrder mode.

IsCalibrationMode [Output] △13-4

A flag is set to determine whether the NetOrder is currently in the calibration mode or not.

Value	Description
QSS_CALIBRAT_ON	NetOrder is in the calibration mode.

QSS_CALIBRAT_OFF		NetOrder is not in the calibration mode.
EnableOutMediaViewer		[Output] △18-5
Available type of viewer is defined with bit allocation.		
Value		Description
QSS_MEDIA_VIEWER_QSS		Comply with QSS setting
QSS_MEDIA_VIEWER_NONE		No Viewer
QSS_MEDIA_VIEWER_SIMPLE		Simple Viewer
QSS_MEDIA_VIEWER_DELUXE		Deluxe Viewer
QSS_MEDIA_VIEWER_PICTURECD_5		Picture CD Vol. 5 or earlier
QSS_MEDIA_VIEWER_PICTURECD_6		Picture CD Vol. 6 or forward
Reserve		
Reserved (Unused)		

## QSS\_PRINT\_CHANNEL structure

```
typedef struct _QSS_PRINT_CHANNEL {
    short                ChNo;
    unsigned short       Meishou[11];
    short                Printtype;
    unsigned char        InpMediaType;
    unsigned short       MeishouCph[3][6];
    short                Haba[3];
    short                Mensitu[3];
    short                Feed[3];
    short                WbHaba[3];
    short                SizeRate[3];
    signed short         RokouichiHosei[3];
    short                CvpSw;
    short                FPSw;
    short                IDPSize[3];
    short                IndexHaba[3];
    short                IndexMensitu[3];
    unsigned char        OutMediaSw;
    unsigned short       OutMediaFormat;
    unsigned char        OutMediaInfoQuality;
    unsigned char        OutMediaInfoQualityPer;
    unsigned char        OutMediaInfoSize;
    unsigned char        PaperFitSW;
    unsigned short       EditModeNo;
    unsigned short       Template;
    unsigned char        PapScan120;
    unsigned char        Reserve1;
    unsigned short       OutPrintSw;
    unsigned short       OutIndexSw;
    unsigned short       OutPrintFrame;
    unsigned char        Reserve[20];
} QSS_PRINT_CHANNEL;
```

### Member

ChNo		[Output]
Define the channel number.		
Meishou		[Output]
Define the channel name.		
Printtype		[Output]
Define the type of print as follows:		

Value	Description
QSS_PRINTTYPE_NONE	Undefined
QSS_PRINTTYPE_NORMAL	Normal print
QSS_PRINTTYPE_EDIT	Edit print

QSS_PRINTTYPE_PACKAGE	Package print
QSS_PRINTTYPE_ALBUM	Album
QSS_PRINTTYPE_LONG Δ7	Long length print

InpMediaType [Output]

Define the type of input media as follows:

Value	Description
QSS_INPMEDIA_NONE	Undefined
QSS_INPMEDIA_CL_NEGA	Color negative
QSS_INPMEDIA_BW_NEGA	Black & white negative
QSS_INPMEDIA_CL_POSI	Color reversal
QSS_INPMEDIA_BW_POSI	Black & white reversal
QSS_INPMEDIA_PRN_PHOTO	Capture image
QSS_INPMEDIA_MO	MO
QSS_INPMEDIA_FD	FD
QSS_INPMEDIA_DVD	DVD
QSS_INPMEDIA_CD	CD
QSS_INPMEDIA_ZIP	ZIP
QSS_INPMEDIA_SM	Smart Media
QSS_INPMEDIA_CF	Compact Flash
QSS_INPMEDIA_PCCARD	PC Card
QSS_INPMEDIA_HD	HD
QSS_INPMEDIA_SEPIA	Sepia
QSS_INPMEDIA_BW_OB	Monochrome (orange base)
QSS_INPMEDIA_CTERM Δ7	C Terminal
QSS_INPMEDIA_RDS Δ7	RDS
QSS_INPMEDIA_SD Δ7	SD Card
QSS_INPMEDIA_MS Δ7	Memory Stick
QSS_INPMEDIA_STORAGE Δ7	d-Storage
QSS_INPMEDIA_USB Δ7	USB Flash Memory
QSS_INPMEDIA_XD_CARD Δ12-5	xD-Picture Card
QSS_INPMEDIA_MINI_SD Δ12-6	miniSD Card
QSS_INPMEDIA_MS_DUO Δ12-7	Memory Stick Duo
QSS_INPMEDIA_DVD_ROM Δ14-1	DVD+/-R/RW

MeishouCph [Output]

Define the print name for each type of print.

From the head of the array, Classical, Panoramic, and High-definition sizes are stored in this order.

Haba [Output]

Define the width of the print for each type of print. (unit: 1/10 mm)

From the head of the array, Classical, Panoramic, and High-definition sizes are stored in this order.

Mensitu [Output]

Define the type of paper surface for each type of print.

The range is 1 – 4 for QSS-28, QSS-29 and QSS-30.

From the head of the array, Classical, Panoramic, and High-definition sizes are stored in this order.

Feed [Output]

Define the advance length of paper for each type of print. (unit: 1/10 mm)

From the head of the array, Classical, Panoramic, and High-definition sizes are stored in this order.

WbHaba [Output]

Define the width of the white border for each type of print. (unit: 1/10 mm)

From the head of the array, Classical, Panoramic, and High-definition sizes are stored in this order.

SizeRate [Output]

Define the magnification ratio of the image for each type of print. (unit: %)

From the head of the array, Classical, Panoramic, and High-definition sizes are stored in this order.

RokouichiHosei [Output]

Define the exposure position correction for each type of print. (unit: 1/10 mm)

From the head of the array, Classical, Panoramic, and High-definition sizes are stored in this order.

CvpSw [Output]

Define CVP printing flag as follows:

Value	Description
QSS_CVP_OFF	CVP disenabled
QSS_CVP_ON	CVP enabled

FPSw

[Output]

Define the front print position as follows:

Value	Description
QSS_FP_NONE	Front print will not be made.
QSS_FP_RIGHT	Front print, right justified
QSS_FP_LEFT	Front print, left justified
QSS_FP_CENTER	Front print, center justified

IDPSize

[Output]

Define the type of index print as follows:

From the head of the array, 135 film, 240 film, and storage media are stored in this order.

Value	Description
QSS_INDEX_4R	4R (102mm x 152mm) index print
QSS_INDEX_3HD	3HD (89mm x 158mm) index print
QSS_INDEX_3R	3R (89mm x 127mm) index print
QSS_INDEX_4HD	4HD (102mm x 178mm) index print
QSS_INDEX_3W	3W (89mm x 178mm) index print
QSS_INDEX_5R	5R (127mm x 178mm) index print
QSS_INDEX_3WS	3WS (89mm x 178mm) index print
QSS_INDEX_3HS	3HS (82.5mm x 158mm) index print
QSS_INDEX_6R	6R (152mm x 203mm) index print
QSS_INDEX_6HD	6HD (152mm x 254mm) index print
QSS_INDEX_6W	6W (152mm x 305mm) index print
QSS_INDEX_8RS	8RS (203mm x 254mm) index print
QSS_INDEX_8R	8R (203mm x 305mm) index print
QSS_INDEX_8HD	8HD (203mm x 356mm) index print
QSS_INDEX_CD40 Δ7	CD_40 (120mm x 120mm) index print
QSS_INDEX_CD40A Δ7	CD_40A (89mm x 120mm) index print
QSS_INDEX_CD40B Δ7	CD_40B (102mm x 120mm) index print
QSS_INDEX_3WL Δ7	3WL (89mm x 254mm) index print
QSS_INDEX_3WL_18 Δ7	3WL_18 (89mm x 254mm) index print
QSS_INDEX_4WL_18 Δ8	4WL_18 (102mm x 254mm) index print
QSS_INDEX_12R Δ13-2	12R (305mm x 457mm) index print
QSS_INDEX_CP6_1	Contact index print of 6 x 1 frames
QSS_INDEX_CP6_2	Contact index print of 6 x 2 frames
QSS_INDEX_CP6_3	Contact index print of 6 x 3 frames
QSS_INDEX_CP6_4	Contact index print of 6 x 4 frames
QSS_INDEX_CP6_5	Contact index print of 6 x 5 frames
QSS_INDEX_CP6_6	Contact index print of 6 x 6 frames
QSS_INDEX_CP6_7	Contact index print of 6 x 7 frames
QSS_INDEX_CP4_1	Contact index print of 4 x 1 frames
QSS_INDEX_CP4_2	Contact index print of 4 x 2 frames
QSS_INDEX_CP4_3	Contact index print of 4 x 3 frames
QSS_INDEX_CP4_4	Contact index print of 4 x 4 frames
QSS_INDEX_CP4_5	Contact index print of 4 x 5 frames
QSS_INDEX_CP4_6	Contact index print of 4 x 6 frames
QSS_INDEX_CP4_7	Contact index print of 4 x 7 frames
QSS_INDEX_CP4_8	Contact index print of 4 x 8 frames
QSS_INDEX_CP4_9	Contact index print of 4 x 9 frames
QSS_INDEX_CP4_10	Contact index print of 4 x 10 frames

IndexHaba

[Output]

Define the paper width of index paper. (unit: 1/10 mm)

From the head of the array, 135 film, 240 film, and storage media are stored in this order.

IndexMensitu

[Output]

Define the paper surface of index print.

The range is 1-4 for QSS-28, QSS-29, and QSS-30.

From the head of the array, 135 film, 240 film, and storage media are stored in this order.

OutMediaSw

[Output]

Define the type of output media as follows:

Value	Description
QSS_OUTPMEDIA_NONE	No media output

QSS_OUTPMEDIA_FD	FD
QSS_OUTPMEDIA_CDR	CD-R
QSS_OUTPMEDIA_MO	MO
QSS_OUTPMEDIA_ZIP	ZIP
QSS_OUTPMEDIA_DVD	DVD
QSS_OUTPMEDIA_CF	Compact Flash
QSS_OUTPMEDIA_SM	Smart Media
QSS_OUTPMEDIA_PC	PC Card
QSS_OUTPMEDIA_HD	HD
QSS_OUTPMEDIA_CDRWSYS	CD-R Writing System
QSS_OUTPMEDIA_SD Δ7	SD Card
QSS_OUTPMEDIA_MS Δ7	Memory Stick
QSS_OUTPMEDIA_BRAVO Δ7	Bravo
QSS_OUTPMEDIA_USB Δ7	USB Flash Memory
QSS_OUTPMEDIA_XD_CARD Δ12-8	xD-Picture Card
QSS_OUTPMEDIA_MINI_SD Δ12-9	miniSD Card
QSS_OUTPMEDIA_MS_DUO Δ12-10	Memory Stick Duo
QSS_OUTPMEDIA_DVD_ROM Δ14-2	DVD+/-R/RW

OutMediaFormat [Output]

Define the output format as follows:

Value	Description
QSS_MEDIA_FORMAT_NONE	None
QSS_MEDIA_FORMAT_JPEG	Jpeg
QSS_MEDIA_FORMAT_FPX	FlashPix
QSS_MEDIA_FORMAT_BMP	Bitmap
QSS_MEDIA_FORMAT_TIFF	TIFF

OutMediaInfoQuality [Output]

Define the image quality as follows:

Value	Description
QSS_MEDIA_QUALITY_0	Standard
QSS_MEDIA_QUALITY_1	Quality 1
QSS_MEDIA_QUALITY_2	Quality 2
QSS_MEDIA_QUALITY_3	Quality 3

OutMediaInfoQualityPer [Output]

Define the quality ratio of the image to be saved to media (unit: %)

OutMediaInfoSize [Output]

Define the output size as follows:

Value	Description
QSS_MEDIA_SIZE_NONE	None
QSS_MEDIA_SIZE_1P4	1/4 BASE
QSS_MEDIA_SIZE_1	BASE
QSS_MEDIA_SIZE_4	4 BASE
QSS_MEDIA_SIZE_16	16 BASE
QSS_MEDIA_SIZE_NONE_HS	None (HS)
QSS_MEDIA_SIZE_1P4_HS	1/4 BASE (HS)
QSS_MEDIA_SIZE_1_HS	BASE (HS)
QSS_MEDIA_SIZE_4_HS	4 BASE (HS)
QSS_MEDIA_SIZE_16_HS	16 BASE (HS)

PaperFitSW [Output]

Define the type of paper fitting as follows:

Value	Description
QSS_PF_CUT	Cut
QSS_PF_WHOLE	Overall
QSS_PF_SAME	Real size

EditModeNo [Output]

Define the edit type as follows:

Value	Description
QSS_EDIT_POST_CARD	Postcard
QSS_EDIT_BUSINESS_CARD	Business card
QSS_EDIT_CERTIFICATE_PHOTO	ID photo

QSS_EDIT_MULTI		Multi	
Template			[Output]
Define the template type.			
Bit assignment of template type is as follows (Bit 1: enabled, Bit 2: Disabled):			
(There are cases where multiple templates are selected.)			
	0: C	1: P	2: H
PapScan120			[Output]
(Unused)			
Reserve1			
Reserved (Unused)			
OutPrintSw			[Output]
(Unused)			
OutIndexSw			[Output]
(Unused)			
OutPrintFrame			[Output]
(Unused)			
Reserve			
Reserved (Unused)			

## QSS\_PU\_INFO structure

```
typedef struct _QSS_PU_INFO {
    char        NameC[20];
    char        NameP[20];
    char        NameH[20];
    unsigned short QuantityC;
    unsigned short QuantityP;
    unsigned short QuantityH;
    unsigned short PriceC;
    unsigned short PriceP;
    unsigned short PriceH;
    unsigned long SumC;
    unsigned long SumP;
    unsigned long SumH;
    unsigned long ChargePrice;
    unsigned long IndexPrice;
    unsigned char Reserve[44];
} QSS_PU_INFO;
```

### Member

NameC, NameP, NameH	[Input]
Define the name of each print size to be output on the pricing sheet.	
NameC: Name of Classical print	
NameP: Name of Panoramic print	
NameH: Name of High-definition print	
NOTE: Although you may define up to 19 characters, the number of characters actually printed on pricing sheet depends on the type of QSS model. (e.g. For QSS-28, QSS-29, and QSS-30, maximum number of characters printed is 5.)	
QuantityC, QuantityP, QuantityH	[Input]
Define the number of resultant prints to be output on the pricing sheet.	
QuantityC: Number of resultant Classical print	
QuantityP: Number of resultant Panoramic print	
QuantityH: Number of resultant High-definition print	
When "0" is set the information related to the print size is not printed on the pricing sheet.	
NOTE: The range is 0 – 999.	
PriceC, PriceP, PriceH	[Input]
Define the unit price of each print size to be output on the pricing sheet.	
PriceC: Unit price of Classical print	
PriceP: Unit price of Panoramic print	

PriceH: Unit price of High-definition print

NOTE: The range is 0 – 9999.

SumC, SumP, SumH

[Input]

Define the total amount of each print size to be output on the pricing sheet.

SumC: Total amount of Classical prints

SumP: Total amount of Panoramic prints

SumH: Total amount of High-definition prints

NOTE: The range is 0 – 999999.

ChargePrice

[Input]

Define the base price of a print.

NOTE: The range is 0 – 9999.

IndexPrice

[Input]

Define the unit price of an index print.

NOTE: The range is 0 – 9999.

Reserve

Reserved (Unused)

Remarks:

Below is a sample of how the information listed above is allocated on a pricing sheet made by PU. (The order of information is always Classical > Panoramic > High-definition.)

Name	Q'ty	Unit price	Sum
INPUT	1	ChargePrice	ChargePrice
<u>NameC</u>	<u>QuantityC</u>	<u>PriceC</u>	<u>SumC</u>
<u>NameP</u>	<u>QuantityP</u>	<u>PriceP</u>	<u>SumP</u>
<u>NameH</u>	<u>QuantityH</u>	<u>PriceH</u>	<u>SumH</u>
INDEX	*1 999		*2 999,999
TAX	*3 99.999 %		*4 999,999
Total amount	*5 (999,999)		*6 999,999

\*1: Number of index print (range: 1 – 999)

\*2: Number of index print times unit price (IndexPrice) (range: 0 – 999999)

\*3: Tax rate whose setting is made on QSS (range: 0.000 – 99.999)

\*4: Tax calculated with the tax rate whose setting is made on QSS (range: 0 – 999999)

\*5: Price exclusive of tax (range: 0 – 999999)

\*6: Price inclusive of tax (range: 0 – 999999)

As for tax rate, fractions, decimal point position, grouping symbol, ones whose setting has been made on QSS are applied.

Note:

Be sure to make setting of prices so they will fall in each allocated area on a pricing sheet.

## QSS\_SUM\_INFO structure

```
typedef struct _QSS_SUM_INFO {
    unsigned long    PChC[100];
    unsigned long    PChP[100];
    unsigned long    PChH[100];
    unsigned long    PaperPrint;
    unsigned long    PaperIndex;
    unsigned long    PaperSetup;
    unsigned long    PaperLabel;
    unsigned long    PaperOther;
```

```

    unsigned long    PaperTotal;
    unsigned long    WriteMedia;
    unsigned long    WriteImage;
    unsigned short   DisposalSpec;
    unsigned long    TotalHojyu[9];
    unsigned char     Reserve[42];
} QSS_SUM_INFO;

```

**Member**

PChC [Output]  
PChP [Output]  
PChH [Output]

Define the total number of prints made in each print channel.

In the 0<sup>th</sup> of an array is the total number of prints of CH1, and in the 1<sup>st</sup> is that of CH2. Thus, the total number of prints in CH1 to CH99 is stored in this structure.

In the 99<sup>th</sup> is the total number of prints made from the external input source.

PChC: Total number of prints of Classical print

PChP: Total number of prints of Panoramic print

PChH: Total number of prints of High-definition print

PaperPrint [Output]  
PaperIndex [Output]  
PaperSetup [Output]  
PaperLabel [Output]  
PaperOther [Output]

Define the total number of prints made by QSS.

PaperPrint: Total number of prints in Print Totals

PaperIndex: Total number of prints in Index Print Totals

PaperSetup: Total number of prints in Setup Print Totals

PaperLabel: Total number of prints in Label Totals

PaperOther: Total number of prints of Others

PaperTotal: Total of PaperPrint, PaperIndex, PaperSetup, PaperLabel, and PaperOther.

WriteMedia [Output]

WriteImage [Output]

Total number of storage media to which images have been written by QSS.

WriteMedia: Total number of media

WriteImage: Total number of images written into storage media.

DisposalSpec [Output]

Define the process specification of QSS as follows:

Value	Description
QSS_SPEC_NORMAL	Standard spec.
QSS_SPEC_SM	SM spec.
QSS_SPEC_J	J spec.
QSS_SPEC_EX	EX spec.

TotalHojyuq [Output]

Define the total amount of replenisher solution used on QSS (unit: ml).

The value stored in array varies depending on the process specification of the machine (DisposalSpec). Refer to the table below:

No	QSS_SPEC_NORMAL	QSS_SPEC_SM	QSS_SPEC_J	QSS_SPEC_EX
0	CD	CD-A	_____	CD-
1	BF	BF-A	_____	BF-A
2	STB	STB	_____	STB
3	_____	CD-B	_____	BF-B
4	_____	CD-C	_____	CD-W
5	_____	BF-B	_____	BF-W
6	_____	CD-W	_____	STB-W
7	_____	BF-W	_____	_____
8	_____	STB-W	_____	_____

Reserve

Reserved (Unused)

## QSS\_PROFILE\_INFO structure

```
typedef struct _QSS_PROFILE_INFO {
    unsigned short    DeviceKind;
    unsigned short    PaperWidth;
    unsigned short    Surface;
    unsigned char     Reserve[26];
} QSS_PROFILE_INFO;
```

### Member

DeviceKind [Input]

Define the device whose profile you wish to get as follows:

Value	Description
QSS_PROFILE_MON	Get the profile of the monitor.
QSS_PROFILE_PRN	Get the profile of the printer.

PaperWidth [Input]

Define the paper width whose profile you wish to get. (unit: 1/10 mm)

NOTE: Use this member when you define QSS\_PROFILE\_PRN for DeviceKind.

Surface [Input]

Define the paper surface whose profile you wish to get.

The range is 1 – 4 for QSS-28, QSS-29, and QSS-30.

NOTE: Use this member when you define QSS\_PROFILE\_PRN for DeviceKind.

Reserve  
Reserved (Unused)

## QSS\_PROFILE\_PIPE structure

```
typedef struct pipe_QSS_PROFILE_PIPE
{
    void    (__RPC_FAR * pull)(
        char __RPC_FAR * state,
        unsigned char __RPC_FAR * buf,
        unsigned long esize,
        unsigned long __RPC_FAR * ecount
    );
    void    (__RPC_FAR * push)(
        char __RPC_FAR * state,
        unsigned char __RPC_FAR * buf,
        unsigned long ecount
    );
    void    (__RPC_FAR * alloc)(
        char __RPC_FAR * state,
        unsigned long bsize,
        unsigned char __RPC_FAR * __RPC_FAR * buf,
        unsigned long __RPC_FAR * bcount
    );
    char __RPC_FAR * state;
} QSS_PROFILE_PIPE;
```

### Member

Pull [Input]

Unused. Define NULL.

Push [Input]

Define the pointer to the profile data transfer function implemented by Client.

For the profile data transfer function, implement the logic to transfer the profile data from the memory on the QSS.

The actual profile data transfer is implemented by RPC pipe.

Alloc

[Input]

Define the pointer to the get memory size function implemented by Client.

For the get memory size function, implement the logic to define the memory size to secure the memory area in the image data destination.

State

[Input]

Define the Client specific data.

NOTE: Refer to Microsoft Platform SDK Documentation for more detail about RPC pipe.

---

## QSS\_PRINTER\_ENUM structure

```
typedef struct _QSS_PRINTER_ENUM
```

```
{
    char          Name[20];
    unsigned char Address[4];
    unsigned short Resolut;
    unsigned short SystemInfo;
    unsigned char Reserve[12];
} QSS_PRINTER_ENUM;
```

△18-6

### Member

Name

[Output]

Define the model name of QSS.

Address

[Output]

Define the IP address.

Resolut

[Output]

Define the default resolution of the printer.

SystemInfo

[Output] △18-7

Define the running status of the QSS.

Refer to SystemInfo of the QSS\_PRINTER\_INFO structure for detail.

Reserve

Reserved (Unused)

---

## QSS\_DATETIME structure △2

```
typedef struct _QSS_DATETIME {
    unsigned short Year;
    unsigned short Month;
    unsigned short Day;
    unsigned short Hour;
    unsigned short Minute;
} QSS_DATETIME;
```

### Member

Year

[Output]

Define year (dominical year).

Month

[Output]

	Define month. The range is 1 – 12.	
Day		[Output]
	Define day. The range is 1 – 31.	
Hour		[Output]
	Define hour. The range is 0 – 23.	
Minute		[Output]
	Define minute. The range is 0 – 59.	

## QSS\_ORDER\_HISTORY structure $\Delta 2$

```
typedef struct _QSS_ORDER_HISTORY {
    QSS_DATETIME      ReceiptTime;
    QSS_DATETIME      CompleteTime;
    unsigned short    ReceiptNo;
    unsigned short    Status;
    unsigned short    FrameNum;
    unsigned short    PaperWidth;
    unsigned short    Surface;
    unsigned short    IndexPrintFlg;
    unsigned short    PaperFittingFlg;
    unsigned short    ReceiptFlg;
    unsigned short    OrderNo;
    char              Host[20];
    char              User[20];
    unsigned short    RequestNo;
    unsigned char     Address[6];
    unsigned short    PrintNumC;
    unsigned short    PrintNumP;
    unsigned short    PrintNumH;
    unsigned short    IndexPrintNum;
    unsigned short    MediaTotal;
    unsigned short    OutputPrint;
    unsigned short    OutputMedia;
    unsigned short    CT1MediaOutput;
    unsigned short    CT1OutputMedia;
    QSS_DATETIME      PrintTime;
    unsigned short    PaperWidthB;
    unsigned short    SurfaceB;
    unsigned short    Reserve1[6];
    unsigned hyper    RefId;
    unsigned char     Reserve[8];
} QSS_ORDER_HISTORY;
```

### Member

ReceiptTime		[Output]						
	Defines the receipt time.							
CompleteTime		[Output]						
	Defines the printing completed time.							
ReceiptNo		[Output]						
	Defines the receipt number.							
Status		[Output]						
	Defines the order type as follows:							
<table><tr><th>Value</th><th>Description</th></tr><tr><td>QSS_ORDER_PRINTED      Δ19-2</td><td>Printed order</td></tr><tr><td>QSS_ORDER_NONE        Δ19-3</td><td>Canceled order</td></tr></table>		Value	Description	QSS_ORDER_PRINTED      Δ19-2	Printed order	QSS_ORDER_NONE        Δ19-3	Canceled order	
Value	Description							
QSS_ORDER_PRINTED      Δ19-2	Printed order							
QSS_ORDER_NONE        Δ19-3	Canceled order							
FrameNum		[Output]						
	Defines the total number of frames.							

PaperWidth	[Output]
Defines the paper width (unit: 1/10 mm).	
Surface	[Output]
Defines the paper surface.	
IndexPrintFlg	[Output]
Defines the index size. For detail, refer to IDPSize of <a href="#">WSQSS_PRINT_CHANNEL structure</a> .	
PaperFittingFlg	[Output]
Defines the type of paper fitting. For detail, refer to PaperFitSW of <a href="#">WSQSS_PRINT_CHANNEL structure</a> .	
ReceiptFlg	[Output]
Defines whether or not to issue order sheet.	
Value	Description
QSS_RECEIPT_ON	Issue order sheet.
QSS_RECEIPT_OFF	Not issue order sheet.
OrderNo	[Output]
Defines order number.	
Host	[Output]
Defines host name.	
User	[Output]
Defines user name.	
RequestNo	[Output]
Defines request number.	
Address	[Output]
Defines MAC address.	
PrintNumC	[Output]
PrintNumP	[Output]
PrintNumH	[Output]
Defines number of print.	
IndexPrintNum	[Output]
Defines number of index print.	
MediaTotal	[Output]
Defines number of media to which data is output.	
OutputPrint	[Output]
Defines whether to print or not.	
OutputMedia	[Output]
Defines the type of output media. For detail, refer to OutMediaSw of <a href="#">WSQSS_PRINT_CHANNEL structure</a> .	
CT1MediaOutput	[Output]
Defines media output on CT-1. When the value is NOT 0, it means the media output is performed on CT-1.	
CT1OutputMedia	[Output]
Defines the type of output media used on CT-1. For detail, refer to OutMediaSw of <a href="#">WSQSS_PRINT_CHANNEL structure</a> .	
PrintTime	[Output]
Defines printing start time.	
PaperWidthB	[Output]
Defines paper width (unit: 1/10 mm).	
SurfaceB	[Output]
Defines paper surface.	
Reserve1	
Unused.	

Refld		[Output]
	Defines reference number.	
Reserve		
	Unused.	

## QSS\_FRAME\_PARAM2 structure △9

```
typedef struct _QSS_FRAME_PARAM2 {
    unsigned short    OrderNo;
    unsigned short    FrameNum;
    unsigned short    FrameNo;
    char              FileName[18];
    unsigned long     FileSize;
    unsigned long     ImageFormat;
    unsigned short    PrintSize;
    unsigned short    RepeatNum;
    unsigned short    RepeatPos;
    char              CvpString1[120];
    char              CvpString2[120];
    unsigned short    CvpFlg;
    unsigned short    PaperWidth;
    unsigned short    PaperLength;
    unsigned short    Surface;
    unsigned short    WithBorder;
    unsigned short    PaperFittingFlg;
    unsigned short    ImageXPixels;
    unsigned short    ImageYPixels;
    unsigned short    Reserve1;
    unsigned hyper    Refld;
    unsigned short    SizeRate;
    unsigned short    Rotate;
    short             CenterX;
    short             CenterY;
    unsigned short    TrimStartPointX;
    unsigned short    TrimStartPointY;
    unsigned short    TrimSizeX;
    unsigned short    TrimSizeY;
    unsigned short    TrimUnitSize;
    unsigned short    Save;
    unsigned short    EnablePaperFittingFlg;
    char              FrontPrintString[32];
    unsigned short    FrontPrintFlg;
    unsigned char     Reserve[24];
} QSS_FRAME_PARAM2;
```

### Member

OrderNo		[Input]
	Request number	
	The range is 0 – 65535. When 65535 (0xFFFF) is defined, an order will be added using the reference number as the administration key.	
FrameNum		[Input]
	Define the total number of frames that make up an order.	
	The range is 1 – 9999. △13-5	
FrameNo		[Input]
	Define the frame number.	
	The range is 1 – 9999. △13-6	
FileName		[Input]

	Define the file name of the image to be sent to QSS. (Mainly used for index print) This string should be a maximum of 17 characters and NULL terminated.	
FileSize		[Input]
	Define the file size of the image to be sent to QSS (unit: Byte).	
ImageFormat		[Input]
	Define the format of the image to be sent to QSS. Define one of the formats defined in SupportImageFormat in QSS_PRINTER_STATE structure via QssGetPrinterState. Any image format can be defined for each individual frame.	
PrintSize		[Input]
	Define the print size. For values that can be set, refer to "PrintSize" of QSS_FRAME_PARAM structure.	
RepeatNum		[Input]
	Define the number of repeat prints to be made. The range is 0 – 999. NOTE: When you define 0, the frame will not be printed, but will be printed on the index print. <b>Allowable range is 1 – 9999 with NetOrder API version 2.3.0 and up.</b>	△26-2
RepeatPos		[Input]
	Define where in the CVP printing to include the repeat count (sequential serial number). 0 - 117: The 1 <sup>st</sup> line of CVP 120 – 237: The 2 <sup>nd</sup> line of CVP printing 255: No repeat count included in CVP printing	
CvpString1		[Input]
CvpString2		[Input]
	Define the string to be printed as CVP. CvpString1: String to be printed on the 1 <sup>st</sup> line of CVP printing CvpString2: String to be printed on the 2 <sup>nd</sup> line of CVP printing Set arbitrary Noritsu Character code. The string must be NULL terminated. You may define up to 115 characters, but the number of characters to be printed as CVP printing depends on the feed length of the paper or the QSS model. When it is defined by using RepeatPos so the repeat count will be included in the CVP printing, the value of repeat count will supersede the information that is supposed to be printed to the predetermined position (3 digits). △12-11	
CvpFlg		[Input]
	Determine whether the CVP printing uses QSS data or the value defined in CvpString1 and CvpString2. For values that can be set, refer to "CvpFlg" of QSS_FRAME_PARAM structure.	
PaperWidth		[Input]
	Define the paper width to be printed. (unit: 1/10 mm) When converting the paper width/length from inch to 1/10mm, refer to "inch – 1/10mm Conversion Table" enclosed in this document. You may define the same number of paper widths as that can be installed to the connected QSS. In case of QSS-31, you may define 2 different paper widths for an order. In order to use this parameter, be sure to set QSS_PRINT_SIZE_FREE_C, QSS_PRINT_SIZE_FREE_P, or QSS_PRINT_SIZE_FREE_H in PrintSize.	
PaperLength		[Input]
	Define the paper advance length for each frame (unit: 1/10 mm). When converting the paper width/length from inch to 1/10mm, please refer to "inch – 1/10mm Conversion Table" enclosed in this document. In order to use this parameter, be sure to set QSS_PRINT_SIZE_FREE_C, QSS_PRINT_SIZE_FREE_P, or QSS_PRINT_SIZE_FREE_H in PrintSize.	
Surface		[Input]
	Define the surface type of the paper to use. The range is 1-4. In order to use this parameter, be sure to set QSS_PRINT_SIZE_FREE_C, QSS_PRINT_SIZE_FREE_P, or QSS_PRINT_SIZE_FREE_H in PrintSize.	
WithBorder		[Input] △18-10
	Define the size of the white border on the finished print. The range is 0-99. (Unit: 1/10 mm)	
PaperFittingFlg		[Input] △23-13
	Specify the paper fitting type. For the detail, refer to "PaperFittingGlg" in the QSS_ORDER_PARAM structure.	△23-7

ImageXPixels	Unused		
ImageYPixels	Unused		
Reserve1	Unused		
RefId		[Input]	
Reference number. You may define any 64-bit identifier. Setting 0xFFFF to Request number (OrderNo) enables to add an order based on the reference number. This also enables to use 64-bit data as administration key in case Client manages orders. You may input any number from 1 to 999999999999999999 (19 digits).			
SizeRate	Unused		
Rotate		[Input]	△15-7
Define the desired rotation angle in increments of 0.1 degree so the image is oriented in the desired angle on the finished print. The range is 0 – 3599. NOTE: With R2R machines, Rotate of the QSS_FRAME_PARAM structure is to be selected from 0, 90, 180, and 270 degree. With QSS, however, the desired rotation angle can be defined in increments of 0.1 degree to achieve more accurate adjustment.			
CenterX	Unused		
CenterY	Unused		
TrimStartPointX		[Input]	△15-8
Define the cropping start position of the input image in the horizontal direction. Unit can be defined in TrimUnitSize. Refer to fig. 5 below for how to specify. NOTE: When the input image is in portrait, it will be necessary to specify TrimStartPointX assuming that the image is rotated 270 degree.			
TrimStartPointY		[Input]	△15-9
Define the cropping start position of the input image in the vertical direction. Unit can be defined in TrimUnitSize. Refer to fig. 5 below for how to specify. NOTE: When the input image is in portrait, it will be necessary to specify TrimStartPointY assuming that the image is rotated 270 degree.			
TrimSizeX		[Input]	△15-10
Define the cropping size of the input image in the horizontal direction. Unit can be defined in TrimUnitSize. Refer to fig. 5 below for how to specify. NOTE: When the input image is in portrait, it will be necessary to specify TrimSizeX assuming that the image is rotated 270 degree.			
TrimSizeY		[Input]	△15-11
Define the cropping size of the input image in the vertical direction. Unit can be defined in TrimUnitSize. Refer to fig. 5 below for how to specify. NOTE: When the input image is in portrait, it will be necessary to specify TrimSizeY assuming that the image is rotated 270 degree.			
TrimUnitSize		[Input]	△15-12
Define the unit for the parameters used for cropping (TrimStartPointX, TrimStartPointY, TrimSizeX, and TrimSizeY) as follows:			
Value		Description	
QSS_TRIM_UNIT_PIXEL		Pixel	
QSS_TRIM_UNIT_PERCENT		Percent	
Save		[Input]	△18-11
Define whether the image will be actually written to the media or not as follows: QSS_OUTPMEDIA_NONE set in OutMediaFlg of the QSS_ORDER_PARAM2 structure supersedes this setting.			

Value	Description
QSS_SAVE_ON	Image will be written to the media.
QSS_SAVE_OFF	Image will not be written to the media.

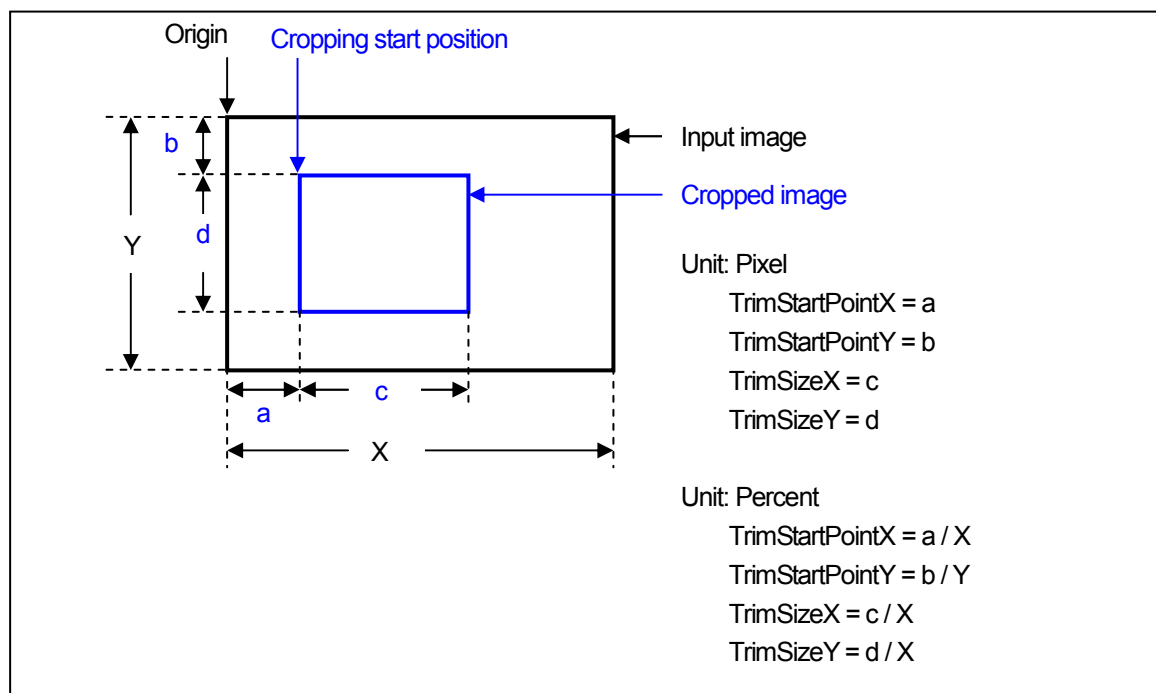


Fig. 5 Cropping of the input image

NOTE: When the input image is in portrait, it will be necessary to specify the start position and the size of the cropping assuming that the image is rotated 270 degree.

EnablePaperFittingFlg [input]  $\Delta$ 23-14

To specify paper fitting type to each individual frame (PaperFittingFlg), set "0". Otherwise, set "1".

FrontPrintString [input]  $\Delta$ 24-13

Specify the character string for front print.

Set arbitrary Noritsu character code. The string must be NULL terminated.

You may define up to 31 characters, but the number of characters to be printed as CVP depends on the advance length of the paper or the QSS model.  $\Delta$ 25-1

FrontPrintFlg [input]  $\Delta$ 24-14

Define the front print position as follows:

Value	Description
QSS_FP_NONE	Front print will not be printed.
QSS_FP_RIGHT	Front print will be right justified.
QSS_FP_LEFT	Front print will be light justified.
QSS_FP_CENTER	Front print will be center justified.

Reserve

Reserved (Unused)

## QSS\_ORDER\_PARAM2 structure $\Delta$ 9

```
typedef struct _QSS_ORDER_PARAM2 {
    unsigned short    OrderNo;
    unsigned short    FrameNum;
    unsigned short    PaperWidth;
    unsigned short    PaperLengthC;
```

unsigned short	PaperLengthP;	
unsigned short	PaperLengthH;	
unsigned short	Surface;	
unsigned short	WithBorderC;	
unsigned short	WithBorderP;	
unsigned short	WithBorderH;	
unsigned short	IndexPrintFlg;	
unsigned short	PaperFittingFlg;	
unsigned short	IndexPaperWidth;	
unsigned short	IndexSurface;	
unsigned short	CmsFlg;	
unsigned short	OrderPunch;	// (Unused)
unsigned hyper	RefId;	
unsigned short	ManualCut;	// (Unused)
char	Comment[22];	△24-15
unsigned short	SorterNum;	
unsigned short	PaperWidthB;	
unsigned short	SurfaceB;	
unsigned short	PaperWidthC;	
unsigned short	SurfaceC;	
unsigned short	IndexPrintNum;	△16-1
unsigned short	OutMediaFlg;	△16-2
unsigned short	OutMediaFormat;	△16-3
unsigned short	OutMediaNum;	△16-4
unsigned short	OutMediaQualityType;	△16-5
unsigned short	OutMediaQuality;	△16-6
unsigned short	OutMediaSize;	△16-7
unsigned short	OutMediaViewer;	△16-8
unsigned short	LabelIndexPrintFlg;	△16-9
unsigned short	LabelIndexNum;	△16-10
unsigned short	LabelIndexPaperWidth;	△16-11
unsigned short	LabelIndexSurface;	△16-12
unsigned short	EnablePriority;	△21-1
unsigned short	Priority;	△16-13
unsigned short	PrintMode;	△16-14
unsigned short	Wait;	△16-15
unsigned short	PaperWidthD;	△24-3
unsigned short	SurfaceD;	△24-4
unsigned char	Reserve[146];	△21-2 △24-5
} QSS_ORDER_PARAM2;		

**Member**

OrderNo	[Input]
Define the Request number. The range is 0 – 65535. When 65535 (0xFFFF) is defined, an order will be added using the reference number as the administration key.	
FrameNum	[Input]
Define the total number of frames that make up an order. The range is 1 – 9999.	
PaperWidth	[Input] △13-7
Define the width of the paper to be printed. (unit: 1/10 mm) When converting the paper width/length from inch to 1/10mm, refer to “inch – 1/10mm Conversion Table” enclosed later in this document.	
PaperLengthC	[Input]
PaperLengthP	[Input]
PaperLengthH	[Input]
Define the paper advance length. (unit: 1/10 mm) PaperLengthC: Paper advance length for Classical size. PaperLengthP: Paper advance length for Panoramic size.	

PaperLengthH: Paper advance length for High-definition size. NOTE: Get the paper information (QSS_PAPER_INFO) with QssGetPaper function, and define the paper feed length between the minimum and maximum feed lengths (PaperLengthMin and PaperLengthMax).		
Surface	[Input]	
Define the surface type of the paper to be printed. The range is 1-4.		
WithBorderC	[Input]	
WithBorderP	[Input]	
WithBorderH	[Input]	
Define the width of the white border on the resultant print. (unit: 1/10 mm) The range is 0 – 99. When 0 is set, the resultant print will be borderless (no white border). WithBorderC: Define the width of the white border for Classical size print. WithBorderP: Define the width of the white border for Panoramic size print. WithBorderH: Define the width of the white border for High-definition size print.		
IndexPrintFlg	[Input]	
Define the paper size of the index print. Number of frames to be printed on an index print will be calculated based on the index print size and number of frames within an order.		
PaperFittingFlg	[Input]	
Define the type of paper fitting to apply to the image, based on the print size. For values that can be set, refer to “PaperFittingFlg” of QSS_ORDER_PARAM structure.		
IndexPaperWidth	[Input]	
Define the width of index print. (unit: 1/10mm)		
IndexSurface	[Input]	
Define the paper surface of the index print. The range is 1-4 for QSS-28, 29, and 30 series.		
CmsFlg	[Input]	
Define whether CMS of QSS will be applied to the order accepted. For values that can be set, refer to “CmsFlg” of QSS_ORDER_PARAM structure.		
OrderPunch		
Unused.		
RefId	[Input]	△1
Reference number. You may define any 64-bit identifier. Setting 0xFFFF to Request number (Order No) enables to add an order based on the reference number. This also enables to use 64-bit data as administration key in case client manages orders. You may input 1 – 999999999999999999 (19 digits).		
ManualCut		
Unused.		
Comment	[Input]	△24-16
Specify the string for Comment. The string must be NULL terminated.		
SorterNum	[Input]	△5
You may define how many prints are placed on a receiver of the sorter before sorter moves. The range is 0 – 120. When you define 0, the sorter will move when the maximum number of prints that a receiver can hold is placed on a receiver. SorterNum is only available with the NetOrder API of version 1.0.6 and on. With the earlier version of NetOrder API, this value is fixed to 0. Client is requested to set the version of the NetOrder API that it uses to Version of QSS_CLIENT_INFO. When the version in use is 1.0.6, set 0x01000600 to Version.		
PaperWidthB	[Input]	
SurfaceB	[Input]	
PaperWidthB: Define the width of the paper to be printed. (unit: 1/10 mm) When converting the paper width/length from inch to 1/10mm, refer to “inch – 1/10mm Conversion Table” enclosed later in this document. SurfaceB: Define the surface type of the paper to be printed. The range is 1-4. When 2 different papers are to be used within an order, define the width and surface type of the second paper.		

Set 0 in the following cases:

Only 1 paper is used for an order.

The connected QSS is of single-magazine type.

PaperWidthC

[Input]

SurfaceC

[Input]

PaperWidthC: Define the width of the paper to be printed. (unit: 1/10 mm)

When converting the paper width/length from inch to 1/10mm, refer to "inch – 1/10mm Conversion Table" enclosed later in this document.

SurfaceC: Define the surface type of the paper to be printed. The range is 1 – 4.

When 3 different papers are to be used within an order, define the width and surface type of the third paper.

Set 0 in the following cases:

Only 1 or 2 paper(s) is(are) used for an order.

The connected QSS is of single- or double-magazine type.

IndexPrintNum

[Input]

△16-16

Define the number of index print to be created. The range is 1 – 999.

IndexPrintNum is only available with the NetOrder API version 2.1.0 or forward. When the client uses the earlier version of NetOrder API, then this value will be ignored and fixed to 1.

When using IndexPrintNum, be sure to set a value bigger than 0x02010000 in Version of QSS\_CLIENT\_INFO. △20-2

OutMediaFlg

[Input]

△16-17

Define the destination of output to media.

For the possible values, refer to OutMediaSw of the QSS\_PRINT\_CHANNEL structure.

When QSS\_OUTPMEDIA\_NONE is defined, output to media will not be performed.

OutMediaFormat

[Input]

△16-18

Define the media output format.

For the possible values, refer to OutMediaFormat of the QSS\_PRINT\_CHANNEL structure.

When QSS\_OUTPMEDIA\_NONE is defined in OutMediaFlg, the value specified here will be ignored.

OutMediaNum

[Input]

△16-19

Define the number of media to be created. The range is 1 – 99.

When QSS\_OUTPMEDIA\_NONE is defined in OutMediaFlg, the value specified here will be ignored.

OutMediaQualityType

[Input]

△16-20

Define the media output quality as follows:

When QSS\_OUTPMEDIA\_NONE is defined in OutMediaFlg, the value specified here will be ignored.

This setting is only applicable when OutMediaFormat is set to either QSS\_MEDIA\_FORMAT\_JPEG or QSS\_MEDIAFORMAT\_FPX. △20-3

Value	Description
QSS_MEDIA_QUALITY_STANDARD	QSS's "Standard"
QSS_MEDIA_QUALITY_Q1	QSS's "Quality 1"
QSS_MEDIA_QUALITY_Q2	QSS's "Quality 2"
QSS_MEDIA_QUALITY_Q3	QSS's "Quality 3"
QSS_MEDIA_QUALITY_SET	Quality specified by client with OutMediaQuality.
QSS_MEDIA_QUALITY_FIXED	Fixed value that QSS internally has.

OutMediaQuality

[Input]

△16-21

Specify the media output quality in percent (%). The range is 1 – 99.

When OutMediaFlg is set to QSS\_OUTPMEDIA\_NONE, the value specified here will be ignored.

This setting is only applicable when OutMediaFormat is set to either QSS\_MEDIA\_FORMAT\_JPEG or QSS\_MEDIAFORMAT\_FPX and when OutMediaQualityType is set to QSS\_MEDIA\_QUALITY\_SET.

△20-4

OutMediaSize

[Input]

△16-22

Define the media output size.

For the possible values, refer to OutMediaInfoSize of the QSS\_PRINT\_CHANNEL structure.

When OutMediaFlg is set to QSS\_OUTPMEDIA\_NONE, the value specified here will be ignored.

OutMediaViewer

[Input]

△16-23

Specify the type of the viewer to be included when writing data to CD.

For the possible values, refer to EnableOutMediaViewer of the QSS\_PRINTER\_STATE structure.

When OutMediaFlg is set to QSS\_OUTPMEDIA\_NONE, the value specified here will be ignored.

This setting is only applicable when OutMediaFlg is set to QSS\_OUTPMEDIA\_CDR, QSS\_OUTPMEDIA\_CDRWSYS, or QSS\_OUTPMEDIA\_BRAVO. △20-5

LabelIndexPrintFlg [Input] △16-24

Specify whether or not to perform label index printing as follows:

When OutMediaFlg is set to QSS\_OUTPMEDIA\_NONE, or when media that QSS label index print function does not support is selected, the value specified here will be ignored. △20-6

Value	Description
QSS_LABEL_OFF	Label index printing will not be performed.
QSS_LABEL_ON	Label index printing will be performed.

LabelIndexPrintNum [Input] △16-25

Define the number of label index print to create. The range is 1 – 99.

When LabelIndexPrintFlg is set to QSS\_LABEL\_OFF, the value specified here will be ignored.

LabelIndexPaperWidth [Input] △16-26

Define the width of the paper on which label index will be printed. (Unit: 1/10 mm)

When LabelIndexPrintFlg is set to QSS\_LABEL\_OFF, the value specified here will be ignored.

LabelIndexSurface [Input] △16-27

Define the surface type of the paper on which label index will be printed. The range is 1 – 4.

When LabelIndexPrintFlg is set to QSS\_LABEL\_OFF, the value specified here will be ignored.

EnablePriority [Input] △21-3

Set 0 when Priority is not used, and 1 when used.

Priority [Input] △16-28

Specify the priority of the order. The range is 0 – 65535.

Priority is only available with NetOrder API version 2.1.0 or forward. When the client uses an earlier version of NetOrder API, then the value specified here will be ignored and fixed to “Normal” priority (QSS\_PRIORITY\_NORMAL).

When Priority is used, set a value bigger than 0x02010000 in Version of QSS\_CLIENT\_INFO and 1 in EnablePriority. When EnablePriority is not set to 1, Priority will be ignored. △21-4

With QSS, values are rounded as follows: △20-7

Value	Possible value	Description
QSS_PRIORITY_HIGHEST	0 – 9	Highest priority
QSS_PRIORITY_HIGH	10 – 149	High priority
QSS_PRIORITY_NORMAL	150 – 9999	Normal
	△22-1	
QSS_PRIORITY_LOW △22-2	10000 – 65534	Low priority
QSS_PRIORITY_NONE △22-3	65535	No priority specified

PrintMode [Input] △16-29

Define the printing method as follows:

Value	Description
QSS_PRINT_MODE_AUTO	AUTO
QSS_PRINT_MODE_PJP	PJP
QSS_PRINT_MODE_PPI	PPI

Wait [Input] △16-30

Define whether or not to suspend processing of order when receiving another order as follows:

Value	Description
QSS_WAIT_OFF	Upon completion of receiving an order, the order state will be set to “Print queue”. (When print data is completely received, printing will start as far as it is ready for printing.)
QSS_WAIT_ON	Upon completion of receiving an order, the order state will be set to “Order reserved”.

PaperWidthD [Input] △24-6

SurfaceD [Input] △24-7

PaperWidthD: Specify the width of the page to print. (unit: 1/10 mm)

For conversion from inch to mm, refer to "inch – 1/10mm Conversion Table" found later in this document.

SurfaceD: Specify the surface type of the paper to use.

You can set a value from 1 to 4.

In order to print an order using 4 different types of paper, set the paper width and surface type of the 4<sup>th</sup> paper.

Set "0" in case of the following:

- Three or less types of paper are used for printing within in an order.
- QSS's of single magazine, double magazine, and triple magazine.

Reserve

Reserved (Unused)

Remarks:

~~For PaperWidth, Surface, and IndexPaperWidth, define the corresponding values of the registered paper.~~

Set values of registered paper to:

PaperWidth, Surface, IndexPaperWidth, IndexSurface, PaperWidthB, SurfaceB, PaperWidthC, SurfaceC, LabelIndexPaperWidth, LabelIndexSurface, PaperWidthD, and SurfaceD △24-8

You may get the information on registered paper by calling QssGetPaper.

PaperWidth, Surface, IndexPaperWidth, IndexSurface, PaperWidthB, SurfaceB, PaperWidthC, SurfaceC, LabelIndexPaperWidth, LabelIndexSurface, PaperWidthD and SurfaceD are used to check to make sure the corresponding paper magazines are installed when printing the order that are spooled in the QSS. △24-9

Note:

On the single-magazine type QSS, types of paper for regular print (PaperWidth and Surface) and index print (IndexPaperWidth and IndexSurface) must be identical.

## Appendix 1: inch – 1/10mm Conversion Table

Use this table for your reference when converting the paper width and/or length from inch to 1/10 mm or vise versa.

inch	1/10 mm
3 1/4	825
3 1/2	890
4	1020
4 1/2	1140
4 5/8	1170
4 3/4	1200
5	1270
5 1/8	1300
6	1520
6 1/2	1650
7	1780
8	2030
8 1/4	2100
8 1/2	2160
9 1/2	2400
10	2540
11	2790
12	3050

## Appendix 2: Noritsu Character Code Table $\Delta 12-12$

Use the following control codes to switch SI and SO codes. For example, to print the copyright mark in SO code, set the following 3 codes in the print data: 0x0e, 0xC1, and 0x0f.

Control code	Description
0x0F	Switch to SI code.
0x0E	Switch to SO code.
0x0D	Switch to SI code (double-size character).
0x0C	Switch to SO code (double-size character).

Table 1: Noritsu Character Code Table (SI code)

Upper Lower		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0 0 0 0	0			SP	0	@	P	`	p				~	タ	ミ		
0 0 0 1	1			!	1	A	Q	a	q				。	ア	チ	ム	
0 0 1 0	2			"	2	B	R	b	r				!	イ	ッ	メ	
0 0 1 1	3			#	3	C	S	c	s				!	ウ	テ	モ	
0 1 0 0	4			\$	4	D	T	d	t				,	エ	ト	ヤ	
0 1 0 1	5			%	5	E	U	e	u				-	オ	ナ	ユ	
0 1 1 0	6			&	6	F	V	f	v				ヲ	カ	ニ	ヨ	
0 1 1 1	7			'	7	G	W	g	w				ヲ	キ	ヌ	ラ	
1 0 0 0	8			(	8	H	X	h	x				イ	ク	ネ	リ	
1 0 0 1	9			)	9	I	Y	i	y				ウ	ケ	ノ	ル	
1 0 1 0	A			*	:	J	Z	j	z				エ	コ	ハ	レ	
1 0 1 1	B			+	;	K	[	k	{				オ	リ	ヒ	ロ	
1 1 0 0	C			,	<	L	\	l					ヤ	シ	フ	ワ	
1 1 0 1	D			-	=	M	]	m	}				ユ	ス	ヘ	ン	
1 1 1 0	E			.	>	N	^	n	~				ヨ	セ	ホ	ス	
1 1 1 1	F			/	?	O	_	o					ッ	ソ	マ	・	

Table 2: Noritsu Character Code Table (SO code)

Upper Lower	0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0 0 0 0 0	0		SP	À	à	Ĭ	ĭ									
0 0 0 0 1	1		ı	Ä	ä	İ	ı						©			
0 0 1 0 0	2		ı	Ä	ä	İ	ı							SP		
0 0 1 0 1	3		~	Ä	ä	Ñ	ñ							-		
0 1 0 0 0	4		£	Á	á	Ö	ö									
0 1 0 0 1	5		-	Á	á	Ö	ö									
0 1 1 0 0	6		—	Æ	æ	Ö	ö									
0 1 1 0 1	7		·		ß	Ó	ó									
1 0 0 0 0	8		·	Ç	ç	Ô	ô									
1 0 0 0 1	9		/	Œ	œ	Ø	ø									
1 0 1 0 0	A		«	Ð	ð	Þ	þ									
1 0 1 0 1	B		»	Ê	ê	Û	û									
1 1 0 0 0	C		§	Ê	ê	Û	û									
1 1 0 0 1	D		»	É	é	Ú	ú									
1 1 1 0 0	E			Ê	ê	Û	û									
1 1 1 0 1	F			İ	ı	μ	€									

## Glossary

### Auto Film Carrier (AFC)

A device to scan all the frames in a strip of film once inserted.

### Auto print (AUTO)

Automatic printing. A type of printing method on QSS.

No image is displayed on the display monitor, and printing does not require operator intervention. All the operator has to do is specify the number of copies to be made and insert film.

### BF

Bleach Fix (process solution)

### CD

Color Developer (process solution)

### Color management

Control color of print through printing process, and produce the stable color on the resultant print.

### Control Strip

A strip that should be processed before starting the printing operation in order to check the condition of process solution.

### CVP

Correction Value Print unit

A device to print the print data onto the backside of print.

### Daily setup

Daily setup compensates for the deterioration of the exposure lamp and process solution so the QSS can constantly produce prints of good quality. It has to be carried out every day before starting the print operation.

### Index print

A print (or sometimes multiple prints depending on how many frames are in the order) that contains all the frames in an order.

### Input from external source

Unlike printing from film, the print request is sent to QSS from the outside of QSS (e.g. From Client via Network.)

### Monthly setup

Setup procedure to be performed monthly.

Adjust the display monitor and update the monitor profile data.

“Execute the Monthly Setup” message is shown on the predetermined date of every month.

### Order

Unit to request and control a series of printing.

In case of film, an order is equivalent to a strip of film, while in case of media, an order is equivalent to a storage media, regardless of the number of frames contained.

**Paper fitting**

Process to match the size of the image data to the paper size.

You can select either Overall, Cut, or Real size.

Overall: The image may be resized so the whole image is printed on the paper. There is a possibility that the resultant print has white border on end.

Cut: The image may be cropped in order to eliminate the presence of white border. Still maintain the aspect ratio of the image, so there is a possibility that outer area of an image may not be printed.

Real size: Image is not resized.

**Paper magazine**

A cassette to be installed in/on the QSS in order to supply paper to the QSS.

A roll of paper to be used to make prints is loaded in the cassette, and the roll has to have its emulsion side facing out.

The number of paper magazine that can be installed on the QSS depends on the QSS model.

**Paper surface**

e.g. Silk, Glossy, etc.

You may register up to 4 paper surfaces to a QSS.

**PJP**

Pre-Judge Printing. A type of printing method on QSS.

Enables operator to make correction of color and density of images and/or make setting of the number of copies while looking at the images on the display monitor.

**PPI**

Acronym of Pre-Print Inspection. A type of printing method on QSS.

Although in AUTO mode the operator does not have a chance to see the images on the monitor, in PPI mode the images are displayed on the monitor for the predetermined period of time so the operator can make correction to the images that need color/density corrections, like PJP.

After the images are displayed for a certain period of time, printing of the images will start automatically in case no corrections are made.

**Print channel**

Each print channel includes the information required for printing like the ones below:

Print name, Print size, Width of white border, Magnification ratio, Exposure position correction, CVP, Front print, Type of index print, Paper magazine for index print, Type of output media, Output format, Image quality, Output size, etc.

**Print size**

There are the following three (3) types of print size:

C: Classic, P: Panoramic, and H: High-definition.

You may assign each frame in an order to the desirable print size listed above.

**Profile data**

A table that describes color reproduction characteristics.

**PU**

Pricing Unit.

Calculate the price and issue a pricing sheet for every order.

**Spillover**

Difference between the image size exposed on paper and the paper size.

Generally set to 1 mm in both horizontal and vertical directions on the full-digital models such as QSS-28 and newer models.

**Spool**

A function to reduce the wait time of CPU and to improve the efficiency of the whole system by using the external memory device as a buffer when inputting/outputting data between the CPU and the peripheral devices in a computer.

**STB**

Stabilizer (process solution)

**Weekly setup**

Regular check to be performed every week and its items are the same as the daily setup.

When you start up the machine “Execute the Weekly setup” is displayed instead of “Execute the Daily setup” on the predetermined date of a week.

**White border**

White area that presences on 4 ends of a print.

There are two types of print – With border and Borderless

## **Trademarks**

Windows 95, Windows 98, Windows 98SE, Windows Me, Windows NT4.0, Windows 2000 are trademarks of Microsoft Corporation.

Macintosh is the registered trademark of Apple Computer, Inc.

NORITSU and QSS are trademarks of Noritsu Koki Co., Ltd.