PMS.PrintProcessor.Service - v3.2.1.0

Solution	Source location
PMS.PrintProcessor.Service	Documentation.md

Table of contents

Introduction

The PMS.PrintProcessor can be used to print labels which are designed by TFormer and filled with information based on an external source (DB or WCF). The Printing can be made on a real printer or only as PDF on disk. The PMS.PrintProcessor has a possibility to print RFID labels based on ZPL. The configuration can be made with the PMS.PrintProcessor.Configurator or in come cases also via Web API.

Configuration

The Documentation of the configuration focus the configuration with the PMS.PrintProcessor.Configuration program. Expert settings are marked as [Expert].

General

Basic settings

Print Processor Name

The Print Processor Name is used for the database communication. So this setting is only relevant if the database interface is activated (see section Database). The both database interfaces had a column where the Print Processor Name is written. If the configured Print Processor name in the PMS.PrintProcessor.Service match the value in the column of the database interface the print requested will be processed by the PMS.PrintProcessor.Service.

Be careful during the configuration that not 2 PMS.PrintProcessor.Services are using the same database with the same Print Processor Name configuration.

Advanced settings [Expert]

Cancel blocked print job after x sec [Expert]

The time period the PMS.PrintProcessor.Service for the completion of a print job. The time is staring when the prepared print job is transmitted to the printer and ends if the printer except the transmission. All preparations for the print job are not included in the time span. Also the time for printing is not included.

Report content of rfid generation and custom serial regardless to the result of printing [Expert]

This flag indicates if the report of TFormer based generated data like RFID content or custom serial number should be also reported to the database interface in case that the printing was not successful or canceled.

File path setting [Expert]

Use Print Processor Individual Temp Path [Expert]

The temp folder of the PMS.PrintProcessor.Service is located in

```
%ProgramFiles(x86)%/MAR GmbH/PMS.PrintProcessor.Service/temp
```

The temp folder is used for content which is temporary needed to store on disk. The temp folder do not contains data which is permanent relevant. (see configuration Cleanup)

In case that the flag is active there is a possibility to change the location of the temp path

IMPORTANT: the path to temp folder must be exists

Use Individual Template Path [Expert]

The template folder of the PMS.PrintProcessor.Service is located in

```
%ProgramFiles(x86)%/MAR GmbH/PMS.PrintProcessor.Service/template
```

In this folders the PMS.PrintProcessor.Service can locate the TFormer templates for printing, the TFormer files for the report of custom serial or other information and also the configuration files for RFID printing in case of using. Files in this folder can be:

- *.tff - TFormer template which can be edit by TFormer
- *.config - RFID configuration file
- *.png - image file which can be used in TFormer template
- *.jpg - image file which can be used in TFormer template

To not hurt the printing don't put test files, backups or subfolder into this folder. Also please don't edit files in this folder. in case of editing copy the labels to another location and edit the files there.

In case that the flag is active there is a possibility to change the location of the template path

IMPORTANT: the path to template folder must be exists

Cleanup [Expert]

Remove files from Temp folder after x days [Expert]

Because of the fact that the temp folder content is not needed for running the PMS.PrintProcessor.Service the content of the folder can be deleted by time. The will made by the time which is configured here.

Database

Connection to database

Use Database interface

If this flag is enabled the PMS.PrintProcessor will try to connect to a configured database to find a Database interface for printing. Actual there are two interfaces possible.

- COM_PRT
  - used for old projects
- PMS_PrintProcessor_1
  - used for new projects

For details to the Database interfaces please take a look into Databases

For connection to database there is a licence for PMS.PrintProcessor.Service needed. This licence must be ordered from Marine- und Automatisierungstechnik Rostock GmbH

- Server
  - The servername where the sql Database is located
- Database
  - The Database name where the Interface is implemented
- Windows Authentication
  - If the Service is running in the context of a domain user the user can be used for the connection. It the flag is off a sql user with password must be configured under user / password
- User
  - this must be a valid sql user with access rights to the database and the interface of the PMS.PrintProcessor.Service
- Password
  - The password related t the configured user. The Password will be saved encrypted in the configuration

Communication Parameters

Skip print job after x min [Expert]

If the createtime of an printjob is older than the actual time minus the configured minutes here the printjob will be ignored.

In case that the print PMS.PrintProcessor.Service is offline this configuration prevent reprint of request's which are to old

Poll Cycle x sec [Expert]

The PMS.PrintProcessor.Service will check the database interface in the cycle which is configured in this configuration. The cycle is stopped during the reading of printjobs from the database. So it is possible that the real cycle is lower than the configuration.

Logging

Log Settings

This configuration handles the logging of the PMS.PrintProcessor.Service.

- Path
  - all logfiles will be located at the configured path. Older files can be find in a subfolder named Archive. In the root there is only the actual file.
- Priority
  - The log priority handle the detail level of logging. For the normal use the level "Warning" is ok. In case that something must be debugged the level can configured lower. If a Loglevel is lower than the configured LogLevel the Logmessage is not written an connect be restored later.

Output

This configuration handles all related settings for the output of an printjob

Printing

Create files only

If this flag is enabled the printing to the requested printer will be ignored. Please use this option only in case that you don't need to print to a real printer and you are only interested in fileprint. In normal use the flag must be off.

Save documents to disk as

The configuration handle the file output format. Most common ist the pdf print.

- PDF
  - Based on the printjob and the template a pdf is generated and stored in the configured output path. The output is normally 100% matching with the real print to a physically printer
- HTML
  - Based on the printjob and the template a html is generated and stored in the configured output path. This output can be differ from the output to a physically printer
- JPG
  - Based on the printjob and the template a jpg is generated and stored in the configured output path. The output is normally 100% matching with the real print to a physically printer
- TXT
  - Based on the printjob and the template a txt is generated and stored in the configured output path. This output differ from the output to a physically printer

Locations

In the locations area you can configure the output path for file printing and the related behavior.

- Use subfolder for each label
  - use this flag to get for each label a subfolder. Normally used in projects where you have a lot of labels and a lot of prinjobs. The subfolder gets the name of the label.
- Report files to Database
  - use this flag is you want to report the pdf output of the labelprint to the database. This configuration can only be used on the old database interface. Regardless of the output configuration the pdf will generate and reported in case that the flag is true

Report based on templates

The PMS.PrintProcessor.Service is able to report generated content based on a label (see Capture Custom Serial Number). In This section you can configure the behavior of reporting this generation results.

- Search option
  - The search option set the deep of searching for labels which have to be reported.
    - AllDirectories

        - All Directories in templetes folder will be used for generation reports
    - TopDirectoryOnly
        - Only the Templete folder and no subdirectores will be used for generation reports

License

In this area you can see your license information in case that you don't have any license or you want to update the license please use the button "Start Licensing".

- Module Name
    - The name of the licensed product. If this case "PMS.PrintProcessor.Service"
- Activation Key
    - The key for activation which is generated for you during the licensing
- Installation ID
    - The id which is related the the license. The installation id is related to the hardware of the pc where the PMS.PrintProcessor.Service is installed. In case that the hardware is changed after the licensing the installation id will be invalid.
- License Key
    - The license key which will send after the purchase of the PMS.PrintProcessor.Service.
- Activated at
    - represents the date and time when the activation was made
- valid until
    - The time when the license will be expired

Network [Expert]

In this area all network parameter can be configured. Possible network connection are the http(s) interface or a WCF interface. Because this configuration is related to another partner the configuration of the network interfaces and the client must match.

Web Interface HTTP [Expert]

Activate and configure a port in case that the http interface is needed to connect another application. Standard use a the connection via the WEB API to configure the PMS.PrintProcessor.Service by web. A valid port must be between 1024 and 65535.

Web Interface HTTPS [Expert]

Activate and configure a port in case that the https interface is needed to connect another application. Standard use a the connection via the WEB API to configure the PMS.PrintProcessor.Service by web. A valid port must be between 1024 and 65535.

For the HTTPS commination a certificate is needed. The location and the password must be configured for a valid https communication

- Certificate path
    - the path to the certificate which should be used for the communication
- Certificate password
    - the password for the certificate which is configured in "Certificate path"

WCF Communication [Expert]

If the communication via wcf should be used the wcf communication must be enabled and the port for the communication must be configured.

WCF metadata exchange [Expert]

If the communication via wcf should be used the wcf metadata exchange must be enabled and the port for the communication must be configured.

RAW IP [Expert]

To communicate with a printer via RAW IP the communication port for RAW IP can be configured. the default port is 9100. The raw IP communication is experimental.

RFID

This section need to be configured in case that the RFID printing is needed for a project. The RFID printing is actual only working for ZEBRA devices. RFID Printing via RAW IP is experimental and not tested. Because of this only the way to print RFID with zebra devices is documented.

The RFID generation is based on the VDA5500.

General RFID structure based on VDA 5500

MB 01: CRC + Protocol Control Word (Header)

Bit Location (Hex)	Data Type	Value	Size	Description
00 – 0F	CRC-16	Hardware assigned	16 bits	Cyclic Redundancy Check
10 – 14	Length	Variable	5 bits	Number of 16-bit words without Protocol Control information and AFI. Check also the table with length below
15	PC bit 0x15	0b0 or 0b1	1 bit	0 = No valid User Data, or no MB 11 / 1 = Valid User Data in MB 11
16	PC bit 0x16	0b0	1 bit	0 = "Extended PC word" not used
17	PC bit 0x17	0b1	1 bit	1 = Data interpretation rules based on ISO/IEC vs GS1
18 – 1F	AFI	e.g.	0xA1, 0xA2	8 bits
	Subtotal		32 bits	

TABLE WITH LENGTH

Bit scheme	length	Bit scheme	length	Bit scheme	length	Bit scheme	length
00000	0	01000	8	10000	16	11000	24
00001	1	01001	9	10001	17	11001	25
00010	2	01010	10	10010	18	11010	26
00011	3	01011	11	10011	19	11011	27

00100	4	01100	12	10100	20	11100	28
00101	5	01101	13	10101	21	11101	29
00110	6	01110	14	10110	22	11110	30
00111	7	01111	15	10111	23	11111	31

MB 01: Unique Item Identifier (UII) with 6 bit encoding

- Start at location 20. Go to end of data /end of available memory

Data Type	Value	Size	Description
Referenz-ID (Platzhalter), Alphanumerisch (an)	user content	n * 6 bits	Reference-ID including Data Identifier (DI)
EoT	0b100001	6 bit	End of Transmission ISO 17363-17367
Padding until the end of the last 16-bit word	0b10, 0b1000, 0b100000, 0b10000010, 0b1000001000, 0b100000100000 or 0b10000010000010	2, 4, 6, 8, 10, 12 or 14 bits	Bit Padding Schema according to ISO/IEC 15962 Chapter 13.1
Subtotal Reference-ID		Variable	96 - 240 bit
Total MB 01		Variable	Up to 272 bits

Codepage 6 bit RFID

Character	Binary Value	Character	Binary Value	Character	Binary Value	Character	Binary Value
Space	100000	0	110000	@	000000	P	010000
EOT	100001	1	110001	A	000001	Q	010001
Reserved	100010	2	110010	B	000010	R	010010
FS	100011	3	110011	C	000011	S	010011
US	100100	4	110100	D	000100	T	010100
Reserved	100101	5	110101	E	000101	U	010101
Reserved	100110	6	110110	F	000110	V	010110
Reserved	100111	7	110111	G	000111	W	010111
(	101000	8	111000	H	001000	X	011000
)	101001	9	111001	I	001001	Y	011001
*	101010	:	111010	J	001010	Z	011010
+	101011	;	111011	K	001011	[	011011
,	101100	<	111100	L	001100	\	011100

-	101101	=	111101	M	001101	]	011101
.	101110	>	111110	N	001110	**GS**	011110
/	101111	?	111111	O	001111	**RS**	011111

Generate RFID from file

In case RFID printing is needed the flag must be enabled.

RFID control file generator

For RFID printing an additional RFID configuration file is needed. This file has the following structure.

```
<?xml version="1.0" encoding="utf-16"?>
<Rfid xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <Epc>
        <PaddingWithZeroes>Right</PaddingWithZeroes>
        <Permalock>true</Permalock>
        <EndOfTransmission>true</EndOfTransmission>
        <Pc>
            <XPC>true</XPC>
            <T>true</T>
            <AFI>A5</AFI>
        </Pc>
    </Epc>
</Rfid>
```

The parameter in the xml file are generated with the xml generator in this section.

XPC

The XPC bit is not defined in the VDA5500 so if there is no reason don't set the this bit to true.

GS1 vs. ISO/IEC

Export of the VDA5500:

> *In the automotive industry two alternative standards for structuring the data contents of RFID transponders have been established:*
>
> - *Data structures according to ISO/IEC principles*
> - *Data structures according to GS1 principles*
>
> *The VDA recommends the implementation of ISO/IEC standards for cross-company RFID applications. However, GS1 standards may be used for intra-company RFID applications. Using GS1 standards in cross-company scenarios requires additional bilateral agreements between the affected supply chain partners.*
>
> *In the following, we describe appropriate ISO/IEC-methods for structuring data in MB 01 and MB 11. GS1 standards are not described in this document. Please review the latest GS1 standards for technical documentation.*

The information if GS1 or ISO/IEC must be activated are based on the information os the customer.

The default case is ISO/IEC.

AFI Code (Application Family Identifier)

The AFI Code depends on the part for which the label is. Possible selectable AFI Codes are in the list below

AFI	Standards
A1	ISO 17367 – Supply chain applications of RFID – Product tagging
A2	ISO 17365 – Supply chain applications of RFID – Transport unit
A3	ISO 17364 – Supply chain applications of RFID – Returnable transport item
A4	ISO 17367 – Supply chain applications of RFID – Product tagging (HazMat)
A5	ISO 17366 – Supply chain applications of RFID – Product packaging
A6	ISO 17366 – Supply chain applications of RFID – Product packaging (HazMat)
A7	ISO 17365 – Supply chain applications of RFID – Transport unit (HazMat)
A8	ISO 17364 – Supply chain applications of RFID – Returnable transport item (HazMat)
A9	ISO 17363 – Supply chain applications of RFID – Freight container
AA	ISO 17363 – Supply chain applications of RFID – Freight container (HazMat)
90	Vehicle identified with the Vehicle Identification Number (VIN) as defined in ISO 3779

Permalock

If the flag permalock is active the epc memory will be locked and cannot be changed in a later process.

EoT - End of transmission

If the flag EoT is enabled the EPC memory string will end with the 6-bit sequence 100001. If the flag is disabled the bit sequence will not add.

Padding

The EPC memory area must be filled up to a complete word. To fill up the epc memory there are actual implement the following possibilities.

Mode	Description	Maximum bit sequence
Left	Filled up to full word from left side with zeros	00 00 00 00 00 00 00
Right	Filled up to full word from right side with zeros	00 00 00 00 00 00 00
Vda5500UmPad	Fill up to full word based on VDA5500 for User Memory	10 00 01 10 00 01 10
Vda5500EpcPad	Fill up to full word based on VDA5500 for EPC	10 00 00 10 00 00 10

Processes

RFID printing

To print a label with a RFID chip the following configurations must be made. The labelname in this example will be 2019:

1. create a label with the name 2019 like a label for normal print which will contains the printed information (2019.tff)
2. create a second label with the same name end the ending _epc. The label can only contains one printable bookmark. Don't use more than one to be sure that the code is generated in the right order (2019_epc.tff)
3. create a configuration file for the label with the same name like the label. Use the PMS.PrintProcessor.Configurator - RFID section to create the config file. (2019.config)

4. activate the Generate RFID from file flag in section "RFID"
5. put all three files into the template folder.
6. test the printing

Generated RFID codes are reported to Database in case that the old Database interface is used with COM_PRT.PrintJob_AddData as xml.

```
<ReportContent>
  <Items>
    <ReportContentEntity>
      <Key>EPC</Key>
      <Value>37SUN5353045051EA971475B*910+0168147!</Value>
    </ReportContentEntity>
  </Items>
</ReportContent>
```

Actual in case of RFID printing the printprocessor is transmitting to database the following informations

Key	Description
EPC	The information of the EPC as user readable information
EPC_HEX	The information of the EPC as HEX
UM	The information of the UM as user readable inforamtion
UM_HEX	The infomation of the UM as HEX

Custom Serial Number

The the old database interface there is a possibility to report data based on a TFormer label. To create such a label the following steps are needed. In the following case the configuration for label 2019 will made.

1. create a additional label with the same name end the ending _report_NAMEOFKEY. The label can only contains one printable bookmark. Don't use more than one to be sure that the code is generated in the right order (2019_report_CustomSerialNumber.tff)
2. if needed a so many labels with reported data as needed e.g. 2019_report_AdditionalData.tff
3. put the files into the template folder
4. test the printing
5. the result should be similar the the following data

```
<ReportContent>
  <Items>
    <ReportContentEntity>
      <Key>CustomSerialNumber</Key>
      <Value>37SUN5353045051EA971475B*910+0168147!</Value>
    </ReportContentEntity>
    <ReportContentEntity>
      <Key>AdditionalData</Key>
      <Value>37SUN5353045051EA971475B*910+0168147!</Value>
    </ReportContentEntity>
  </Items>
</ReportContent>
```

Interfaces

XML Structure

The xml which contains the data for printing can have in maximum the following data.

```
<PrintData>
    <PSData></PSData>
    <LabelName></LabelName>
    <PrintCopies></PrintCopies>
    <DocumentName></DocumentName>
    <DocumentPath></DocumentPath>
    <PrinterName></PrinterName>
    <PrintWithoutPrinter></PrintWithoutPrinter>
    <SaveDocumentsToDiskAsPdf></SaveDocumentsToDiskAsPdf>
    <PrinterType></PrinterType>
    <!-- <RFID>
        <TemplateName></TemplateName>
        <EncryptionKey></EncryptionKey>
        <AccessPassword></AccessPassword>
        <KillPassword></KillPassword>
        <UserMemoryRawData></UserMemoryRawData>
        <Epc>
          <EpcMemoryRawData Permalock=?? PaddingWithZeroes=?? GenerateFromFile=??>
            Text
          </EpcMemoryRawData>
        <Segments>
          <Position></Position>
          <Length></Length>
          <Data></Data>
        </Segments>>
      </Epc>
    </RFID> -->
    <RawIpSequence></RawIpSequence>
</PrintData>
```

- PSData
    - The PSData block contains the data for the tff file. The data is corresponding to the format which is used in TFormer.
- Labelname
    - Name of the label which should be used for printing. A corresponding tff file must be located in the template folder.
- PrintCopies
    - Number of copies of the label. Default is one. In case that the label should be printed more increase the value.
- DocumentName
    - name of the document which should be used in case of saving the print result as file.
- DocumentPath

    - the path where the document should be placed. Override the configured output path.
- PrinterName
    - Name of the printer which should use for the printing
- PrintWithoutPrinter
    - set the flag to true if the printjob should not transfer to printer
- SaveDocumentsToDiskAsPdf
    - set the flag to true if the printjob should be saved as pdf to disk
- PrinterType
    - set the printertype to a valid type
    - valid types are

Printer Type	Value
Default	0
Win32	1
Repository	2
Cups	3
PostscriptFile	4
HtmlFile	5
PdfFile	6
PostscriptPrinter	7
TextPrinter	8
TextFile	9
ePrinterType_ImageBmp	10
ImageBmp	10
ePrinterType_ImageGif	11
ImageGif	11
ePrinterType_ImageJpg	12
ImageJpg	12
ePrinterType_ImagePcx	13
ImagePcx	13
ePrinterType_ImagePng	14
ImagePng	14
ePrinterType_ImageTga	15

ImageTga	15
ePrinterType_ImageTifSingle	16
ImageTifSingle	16
ePrinterType_ImageTifMultiPage	17
ImageTifMultiPage	17
ePrinterType_ZPLPrinter	18
ZPLPrinter	18
ePrinterType_ZPLFile	19
ZPLFile	19
Invalid	20

- RawIpSequence
  - i case that a RAW IP sequence should send to printer fil this parameter
  - NOT RELEASED

WEB API

GetConfiguration

- Method: GET
- Parameter: NONE

This endpoint returns the actual configuration to the calling client like in the example.


```
{
  "DatabaseName": "",
  "DatabaseCatalog": "",
  "DatabaseUser": "",
  "DatabasePassword": "",
  "DatabaseIntegratedSecurity": false,
  "PrintJobSkipOlderThanXMin": 2,
  "PrintJobCancelAfterXSeconds": 10,
  "PrintProcessorName": "PrintProcessor01",
  "NLogLogPath": "C:\\PMS.PrintProcessor.Service\\Log",
  "NLogLogPriority": 1,
  "SaveDocumentsToDatabase": false,
  "SaveDocumentsToDiskAsPdf": false,
  "SaveDocumentsToDiskPath": "C:\\PMS.PrintProcessor.Service\\Output",
  "PrintProcessorPrintWithoutPrinter": false,
  "PrintProcessorStandaloneMode": true,
  "PrintProcessorPollCycleInSec": 5,
  "ReportContentRegardlessToPrintResult": false,
  "SaveDocumentsToDiskAsHtml": false,
  "SaveDocumentsToDiskAsJpg": false,
  "SaveDocumentsToDiskAsTxt": false,
  "PrintProcessorIndividualTempPathActive": false,
  "PrintProcessorIndividualTempPath": "C:\\PMS.PrintProcessor.Service\\Temp",
```

```
  "PrintProcessorTempPathCleanUpDays": 30,
  "PrintProcessorIndividualTemplatePathActive": false,
  "PrintProcessorIndividualTemplatePath": "C:\\PMS.PrintProcessor.Service\\Templates",
  "SaveDocumentsToDiskIntoSubfolderPerLabel": true,
  "LicenseData": "",
  "CertificatePath": "C:\\certificate.cer",
  "CertificatePassword": "",
  "WebInterfaceHttp": false,
  "WebInterfaceHttps": false,
  "WebInterfaceHttpPort": 55000,
  "WebInterfaceHttpsPort": 55001,
  "WcfHost": false,
  "WcfHostPort": 55002,
  "WcfMexPort": 55003,
  "WcfMexUse": false,
  "RawIpPort": 9100,
  "RfidGenerateFromFile": false
}
```


SetConfiguration

- Method: POST
- Parameter: Base.DataTypes.Settings.ConfigModel

The endpoint has the possibility to change the configuration of the PMS.PrintProcessor.Service from remote. The serialized Base.DataTypes.Settings.ConfigModel must be given as parameter.


```
{
  "DatabaseName": "",
  "DatabaseCatalog": "",
  "DatabaseUser": "",
  "DatabasePassword": "",
  "DatabaseIntegratedSecurity": false,
  "PrintJobSkipOlderThanXMin": 2,
  "PrintJobCancelAfterXSeconds": 10,
  "PrintProcessorName": "PrintProcessor01",
  "NLogLogPath": "C:\\PMS.PrintProcessor.Service\\Log",
  "NLogLogPriority": 1,
  "SaveDocumentsToDatabase": false,
  "SaveDocumentsToDiskAsPdf": false,
  "SaveDocumentsToDiskPath": "C:\\PMS.PrintProcessor.Service\\Output",
  "PrintProcessorPrintWithoutPrinter": false,
  "PrintProcessorStandaloneMode": true,
  "PrintProcessorPollCycleInSec": 5,
  "ReportContentRegardlessToPrintResult": false,
  "SaveDocumentsToDiskAsHtml": false,
  "SaveDocumentsToDiskAsJpg": false,
  "SaveDocumentsToDiskAsTxt": false,
  "PrintProcessorIndividualTempPathActive": false,
  "PrintProcessorIndividualTempPath": "C:\\PMS.PrintProcessor.Service\\Temp",
  "PrintProcessorTempPathCleanUpDays": 30,
  "PrintProcessorIndividualTemplatePathActive": false,
  "PrintProcessorIndividualTemplatePath": "C:\\PMS.PrintProcessor.Service\\Templates",
```

```
  "SaveDocumentsToDiskIntoSubfolderPerLabel": true,
  "LicenseData": "",
  "CertificatePath": "C:\\certificate.cer",
  "CertificatePassword": "",
  "WebInterfaceHttp": false,
  "WebInterfaceHttps": false,
  "WebInterfaceHttpPort": 55000,
  "WebInterfaceHttpsPort": 55001,
  "WcfHost": false,
  "WcfHostPort": 55002,
  "WcfMexPort": 55003,
  "WcfMexUse": false,
  "RawIpPort": 9100,
  "RfidGenerateFromFile": false
}
```

Databases

Old Database Interface

Table [COM_PRT].[PrintJob]

The table contains all printjobs which are needed to print from the PMS.PrintProcessor.Service. All labels will be processed which are matching the following criteria.

- The value in PrintProcessor must match the PMS.PrintProcessor.Service name in configuration.
- The CreateTime value is not older than the actual DateTime minus the value in configuration Skip Print Job After to prevent very old prints
- Processtime is empty. The PMS.PrintProcessor.Service will ignore entries with a processtime unequal NULL

The table must have a matching structure

```
CREATE TABLE [COM_PRT].[PrintJob]
(
    [ID]                BIGINT
                        NOT NULL,
    [PrintProcessor]    VARCHAR(50)
                        NOT NULL,
    [XML]               VARCHAR(max)
                        NOT NULL,
    [CreateTime]        DATETIME
                        NOT NULL
                        DEFAULT(GETDATE()),
    [ProcessTime]       DATETIME
                        NULL,
    [ErrorCode]         INT
                        NOT NULL
                        DEFAULT(0),
    ) ON [PRIMARY]
) ON [PRIMARY]
```

Procedure [COM_PRT].[PrintJob_write]

The procedure is called to complete a printjob from table. The following SQL structure is needed.

```
CREATE PROCEDURE [COM_PRT].[PrintJob_write]
    -- ------------------------------------
    -- IDENTITY Parameters
    -- ------------------------------------
    @bint_ID                BIGINT          = NULL OUTPUT,
    -- ------------------------------------
    -- OPTIONAL INPUT Parameters
    -- ------------------------------------
    @str_PrintProcessor     VARCHAR(50)     = NULL,
    @str_XML                VARCHAR(MAX)    = NULL,
    @dt_ProcessTime         DATETIME        = NULL,
    @int_ErrorCode          INT             = NULL,
    @str_NodeName           VARCHAR(30)     = NULL,
    @str_ApplicationName    VARCHAR(30)     = NULL,
    @str_UserName           VARCHAR(30)     = NULL,
    -- ------------------------------------
    -- RETURN Parameters
    -- ------------------------------------
    @bint_ErrorID           BIGINT      = NULL  OUTPUT
BEGIN
```

The PMS.PrintProcessor.Service will fill the parameters with the following data:

- @bint_ID
  - the id which is actual processed
- @str_PrintProcessor
  - in each case NULL
- @str_XML
  - in each case NULL
- @dt_ProcessTime
  - actual date and time
- @int_ErrorCode
  - a error code which is corresponding to the errorcodes in table Database errorcodes
- @str_NodeName
  - the name of the PMS.PrintProcessor.Service host
- @str_ApplicationName
  - in each case "PMS.PrintProcessor.Service"
- @str_UserName
  - The user name of the person who is currently logged on to the Windows operating system and is running the PMS.PrintProcessor.Service
- **OUTPUT** @bint_ErrorID
  - The result of the procedure call. All values unequal 0 are interpreted as error

Procedure [COM_PRT].[Documents_add]

This procedure in the old interface can be used to save the result of the printing as pdf as memory stream in the database.

```
CREATE PROCEDURE [COM_PRT].[Documents_add]
    -- ------------------------------------
    -- IDENTITY Parameters
```

```
    -- ------------------------------------
    @bint_ID                    BIGINT              = NULL,
    -- ------------------------------------
    -- OPTIONAL INPUT Parameters
    -- ------------------------------------
    @bin_Document               VARBINARY(MAX)      = NULL,
    @str_DocumentType           VARCHAR(50)         = NULL,
    @str_NodeName               VARCHAR(30)         = NULL,
    @str_ApplicationName        VARCHAR(30)         = NULL,
    @str_UserName               VARCHAR(30)         = NULL,
    -- ------------------------------------
    -- RETURN Parameters
    -- ------------------------------------
    @bint_ErrorID               BIGINT             = NULL    OUTPUT
BEGIN
```

The PMS.PrintProcessor.Service will fill the parameters with the following data:

- @bint_ID
  - the id which is actual processed
- @bin_Document
  - the pdf as binary stream for the related id
- @str_DocumentType
  - fix value "application/PDF"
- @str_NodeName
  - the name of the PMS.PrintProcessor.Service host
- @str_ApplicationName
  - in each case "PMS.PrintProcessor.Service"
- @str_UserName
  - The user name of the person who is currently logged on to the Windows operating system and is running the PMS.PrintProcessor.Service
- **OUTPUT** @bint_ErrorID
  - The result of the procedure call. All values unequal 0 are interpreted as error

Procedure [COM_PRT].[PrintJob_AddData]

This procedure in the old interface can be used to report the content of an label generation back to the database. The normal usecase is to report a custom serial number or a epc code to the database.

```
CREATE PROCEDURE [COM_PRT].[PrintJob_AddData]
    @bint_JobId             BIGINT,
    @str_PrintProcessor     VARCHAR(50)     = NULL,
    @str_Data               NVARCHAR(MAX),
    @str_NodeName           VARCHAR(100)    = NULL,
    @str_ApplicationName    VARCHAR(100)    = NULL,
    @str_UserName           VARCHAR(100)    = NULL,
    @int_ErrorCode          INT             = NULL OUTPUT,
@str_ErrorText          NVARCHAR(MAX)       = NULL OUTPUT
BEGIN
```

The PMS.PrintProcessor.Service will fill the parameters with the following data:

- @bint_ID

    - the id which is actual processed
- @str_PrintProcessor
    - the print processor which report the content
- @str_Data
    - the xml with the content of reporting (see the example below)
- @str_NodeName
    - the name of the PMS.PrintProcessor.Service host
- @str_ApplicationName
    - in each case "PMS.PrintProcessor.Service"
- @str_UserName
    - The user name of the person who is currently logged on to the Windows operating system and is running the PMS.PrintProcessor.Service
- **OUTPUT** @int_ErrorCode
    - The result of the procedure call. All values unequal 0 are interpreted as error
- **OUTPUT** @str_ErrorText
    - The text result of the procedure call. Empty in case of no error

REPORT EXAMPLE

```
<ReportContent>
  <Items>
    <ReportContentEntity>
      <Key>CustomSerialNumber</Key>
      <Value>37SUN5353045051EA971475B*910+0168147!</Value>
    </ReportContentEntity>
    <ReportContentEntity>
      <Key>AdditionalData</Key>
      <Value>37SUN5353045051EA971475B*910+0168147!</Value>
    </ReportContentEntity>
  </Items>
</ReportContent>
```

New Database Interface

Table [PMS_PrintProcessor_1].[ProcessJob_1]

The table contains all printjobs which are needed to print from the PMS.PrintProcessor.Service. All labels will be processed which are matching the following criteria.

- The value in ProcessorName must match the PMS.PrintProcessor.Service name in configuration.
- The Created value is not older than the actual DateTime minus the value in configuration Skip Print Job After to prevent very old prints

In case that an entry will be completed by the PMS.PrintProcessor.Service the entry must be removed from the table or a related view.

```
CREATE TABLE [PMS_PrintProcessor_1].[ProcessJob_1]
(
    [JobId]             BIGINT          NOT NULL    IDENTITY(1,1),
    [JobName]           VARCHAR(100)    NULL,
    [ItemId]            BIGINT          NULL,
    [ItemName]          VARCHAR(100)    NULL,
    [ProcessorName]     VARCHAR(100)    NULL,
```

```
    [Data]              NVARCHAR(MAX)    NULL,
    [Created]           DATETIME         NOT NULL    DEFAULT(GETDATE())

    CONSTRAINT [PK] PRIMARY KEY([Id])

) ON [PRIMARY]
```

Procedure [PMS_PrintProcessor_1].[CompleteJob_1]

The procedure is called to complete a printjob from table. The following SQL structure is needed.

```
CREATE PROCEDURE [PMS_PrintProcessor_1].[CompleteJob_1]
    @bint_JobId             BIGINT,
    @int_ProcessErrorCode   INT,
    @str_ProcessErrorText   NVARCHAR(MAX)   = NULL,
    @str_NodeName           VARCHAR(100)    = NULL,
    @str_ApplicationName    VARCHAR(100)    = NULL,
    @str_UserName           VARCHAR(100)    = NULL,
    @int_ErrorCode          INT             = NULL  OUTPUT,
    @str_ErrorText          NVARCHAR(MAX)   = NULL  OUTPUT
BEGIN
```

The PMS.PrintProcessor.Service will fill the parameters with the following data:

- @bint_JobId
  - the id which is actual processed
- @int_ProcessErrorCode
  - a errorcode which is corresponding to the errorcodes in table Database errorcodes
- @str_ProcessErrorText
  - Error while Printing + [Error code name] + , check log file.
- @str_NodeName
  - the name of the PMS.PrintProcessor.Service host
- @str_ApplicationName
  - in each case "PMS.PrintProcessor.Service"
- @str_UserName
  - The user name of the person who is currently logged on to the Windows operating system and is running the PMS.PrintProcessor.Service
- **OUTPUT** @int_ErrorCode
  - The result of the procedure call. All values unequal 0 are interpreted as error
- **OUTPUT** @str_ErrorText
  - The result of the procedure call. If @int_ErrorCode unequal 0 this message will be logged

Procedure [PMS_PrintProcessor_1].[SetPrintProcessor_1]

The new PMS.PrintProcessor.Service interface is providing a function to report all printers from the server where the PMS.PrintProcessor.Service is installed to the Database

```
CREATE PROCEDURE [PMS_PrintProcessor_1].[SetPrintProcessor_1]
    @str_ProcessorName      VARCHAR(100),
    @str_Data               NVARCHAR(MAX),
    @str_NodeName           VARCHAR(100)    =    NULL,
    @str_ApplicationName    VARCHAR(100)    =    NULL,
    @str_UserName           VARCHAR(100)    =    NULL,
```

```
    @int_ErrorCode          INT             =    NULL    OUTPUT,
    @str_ErrorText          NVARCHAR(MAX)   =    NULL    OUTPUT
BEGIN
```

- @str_ProcessorName
  - The name of the PMS_PrintProcessor.Service which is reporting the printer
- @str_Data
  - a list with all installed printers based on json

```
[
    {
        "Name": string,
        "Comment": string,
        "Location": string
    },
    {
        "Name": string,
        "Comment": string,
        "Location": string
    }
]
```

- @str_NodeName
  - the name of the PMS.PrintProcessor.Service host
- @str_ApplicationName
  - in each case "PMS.PrintProcessor.Service"
- @str_UserName
  - The user name of the person who is currently logged on to the Windows operating system and is running the PMS.PrintProcessor.Service
- **OUTPUT** @int_ErrorCode
  - The result of the procedure call. All values unequal 0 are interpreted as error
- **OUTPUT** @str_ErrorText
  - The result of the procedure call. If @int_ErrorCode unequal 0 this message will be logged

Database errorcodes

Code	Name
0	NoError
1	GeneralException
2	CanceledByTimeout
10	XmlStructureError
20	XmlToVariableAssignmentError

30	ConvertingError
31	NoPrintCountSet
32	NoDataToPrint
40	TemplateFileNotFound