

Peripheral Input/Output Services

Peripheral Input/Output Services.....	3
PIOS Release Notes.....	3
Peripheral Input/Output Services (New).....	3
Document Generator (new).....	3
Emulator (new)	4
Driver Selection Tool (new)	4
Peripheral Input/Output Services Infrastructure.....	5
PIOS Getting Started	6
PIOS API Core	10
Printer API.....	11
PIOS Printer API Features Description	11
PIOS Printer API Guidelines	13
Printer API Examples	15
Print Text on the Left Paper.....	15
Print Text Center Alignment in Line Mode.....	17
Print Text on the Right Side.....	18
Text Drawn on the Left Side of the Page.....	19
Draw Text in the Center of the Page in Graphic Mode.....	21
Text Drawn on the Right Side of the Page	22
Two Page Report.....	24
Image Printing with Printer in Line Mode.....	25
Printing a Barcode with Printer in Line Mode	27
Text Rotation	29
Rotate an Image	31
Rotating a Barcode.....	33
Configure the Printer Parameters.....	35
Adding/Remove a Font Using the Client API.....	37
PIOS Add-on Drivers	39
Installing a Driver Add-on	39
Driver Configuration	40
MDK Peripheral Support Actions	42
Driver Requirements Document Editor	42
Using the Driver Requirements Document Editor.....	43
MDK Peripheral Support Actions Toolbar	44
Creating a Driver Requirements Document.....	45
Modifying the Driver Requirements Document.....	46
Launching the Peripheral I/O Emulator	46

Set MDK Peripheral Emulation Mode	47
Peripheral Input/Output Emulator	48
PIOS Emulator Menu Options	49
File Menu	49
Edit Menu	50
View Menu	50
Peripheral Options Menu	50
Help Menu	51
Printer Peripheral Panel	51
Emulator Configuration	52
Peripheral Options Menu for Printer Peripheral Type	52
Property Files for Printer Peripheral Type	55
Installing a Font in the Emulator Printer	56
Uninstalling a Font from the Emulator Printer	59
Installing New Media to the Emulator Printer	60
Uninstalling Media from the Emulator Printer	63
Installing a Font Configuration	64
Uninstalling a Font Configuration	67
Using the PIOS Emulator for the Printer Peripheral Type	67
Using the Measuring String	69
Driver Selection Tool	69
Using the Driver Selection Tool	70



Peripheral Input/Output Services

Release Notes

Release Notes only contain a summary of the new features or changes. The corresponding documentation contains detailed information.

Peripheral Input/Output Services (New)

Technical Data

Function is	New
Release	Software Component <ul style="list-style-type: none">• Component: SAP_ME_SERVER_COMPONENT• Release: MI 2.5 SP09
Assignment to Application Component	CA-ME-SER Server-side
Country Setting	Valid for all countries

Use

PIOS provides an abstraction layer between the application and the peripheral.

Document Generator (new)

This is a new component of the SAP NetWeaver Development Studio.

Technical Data

Function is	New
Release	Software Component <ul style="list-style-type: none">• Component: SAP_ME_SERVER_COMPONENT• Release: MI 2.5 SP09
Assignment to Application Component	CA-ME-NTV Native / OS specific drivers
Country Setting	Valid for all countries

Use

The Document Generator is a new MDK component that enables MI developers to specify peripheral requirements within a mobile application project. These requirements are stored on a file called Driver Requirements Document (DRD). The DRD identifies the peripheral types, as well as, the attributes within those peripheral types required by the application. In this first release, the Document Generator supports the printer peripheral type.

Emulator (new)

This is a new component of the SAP NetWeaver Developer Studio.

Technical Data

Function is	New
Release	Software Component <ul style="list-style-type: none">• Component: SAP_ME_SERVER_COMPONENT• Release: MI 2.5 SP09
Assignment to Application Component	CA-ME-SER Server-side
Country Setting	Valid for all countries

Use

The Peripheral I/O Emulator enables the developer to simulate the usage and functionality of peripheral types supported under the PIOS architecture. In this first release, it provides emulation for the printer peripheral type.

Driver Selection Tool (new)

Technical Data

Function is	New
Release	Software Component <ul style="list-style-type: none">• Component: SAP_ME_SERVER_COMPONENT• Release: MI 2.5 SP09
Assignment to Application Component	CA-ME-SER Server-side
Country Setting	Valid for all countries

Use

With release MI 2.5 SP09 the Driver Selection Tool (DST) is a new technical component of the SAP MI Web Console. Administrators need to deploy the appropriate peripheral add-on for mobile applications. The selection process can be very complex due to multiple factors, such as peripheral attributes, operating system, virtual machine, processor, and available transports. These must be taken into consideration when selecting the appropriate driver.

The DST allows the system administrator to easily identify the available peripheral add-ons that match all these requirements.

For more information, see [Driver Selection Tool \[Seite 69\]](#) (SAP Mobile Engine → SAP ME for Administrators → SAP MI Web Console → Driver Selection Tool).

Effects on System Administration

A button that launches the DST is enabled in the Mobile Component page of the SAP MI Web Console.



Peripheral Input/Output Services Infrastructure

Purpose

Services provided by the SAP Mobile Infrastructure to provide peripheral access to mobile applications. PIOS provides an abstraction layer between the application and the peripheral. A developer using PIOS does not have to be concerned about the implementation details of each peripheral model supported. Instead the developer will target abstracted functionality provided by the peripheral and required by the application.

Integration

The PIOS infrastructure has design components, runtime components and an administration component. The design time components are integrated in the SAP NetWeaver Developer Studio. Runtime components and the administration component are integrated in the SAP Mobile Infrastructure Client and the SAP Web Console respectively.

Design-time components:

MDK Peripheral Support Actions - SAP NetWeaver Developer Studio toolbar components used to define and emulate the peripheral features required by an MI application.

- **Create/Modify Driver Requirements Document** - Launches a wizard to specify the mobile application peripheral requirements.
- **Set MDK Peripheral Emulation Mode** - Configures the SAP NetWeaver Developer Studio to run the application peripheral actions/requests in emulation mode.
- **Launch the Peripheral I/O Emulator** - Launches the Peripheral I/O Emulator (described below).

Peripheral Emulator - Emulates peripheral functionality according to settings determined by the developer.

Run-time components:

PIOS Classes in the SAP MI Client - Used to discover, connect, and use functionality available in supported peripheral types.

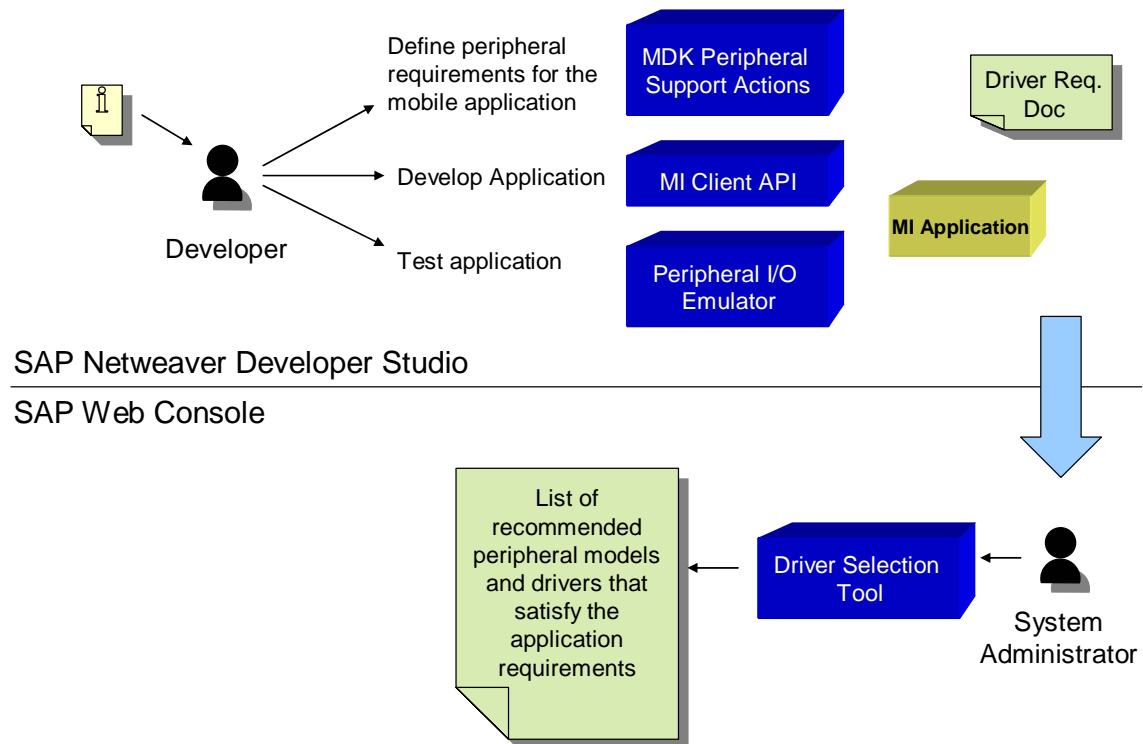
Driver Add-on - Component containing the software modules required to make use of a specific peripheral model on a client device.

Administrative component:

Driver Selection Tool in the SAP Web Console - Identifies available peripheral drivers that match the peripheral requirements of a mobile application (as specified in the Driver Requirements Document).

Example

The following diagram provides an overview of the PIOS infrastructure:



PIOS Getting Started


Use

The procedure explained below provides a quick walkthrough of the PIOS Infrastructure. The process will start by using the Client API to do a simple example, setting the development environment to test the application in the emulator, and finally the application will run in the emulator and the printer.

Prerequisites



- SAP Mobile Infrastructure installed.
- SAP NetWeaver Developer Studio installed and configured.
 - Complete MDK configuration.
 - MI login user name
 - MI login password
- MI Client installed and successfully synchronized.

Procedure

1. Open the SAP NetWeaver Developer Studio.
2. Create a new MI Project by pressing the *MDK: Create a new MI project*  button. Select the *JSP Project* radio button and fill out all the required fields (*Project Name*, *Project root folder*, and so on). Leave the *JSP name* field with the default value (*initial.jsp*). Press *Finish*.



If a browser windows with the XML shows up, close it.

3. Select the project you just created on the *MI Projects* list and then click on  *Create/Modify Driver Requirements Document*.
4. Select *Printer* and then click *Add → Finish*. This will open the **Driver Requirements Document Editor**.
5. On the *Driver Requirements Document Editor* check the boxes for *Bitmapped fonts* and *Scalable fonts*. Click on  *Save*.
6. Completely replace the code in the *initial.jsp*, whose tab is at the top of the DRD Editor, file with the code below:

```
<%@ page import="com.sap.ip.me.api.pios.connection.*" %>
<%@ page import="com.sap.ip.me.api.pios.printer.*" %>

<html>
<head>

<!-- You can place a image here to return to ME home page: 127.0.0.1
= localhost on the Desktop and also works on the PDA
    <a href="http://127.0.0.1:4444/me" > </a>
    -->
</head>
<body>

    <jsp:useBean id="servletToJSPBean" scope="session"
class="miProjectPackage.bean.dataBean" />
<!-- For event handling we need a HTML "form" command -->
    <form method="post" action="start" id="myForm1" name="form">

<!-- Display title of the example -->
    <h4>Hello World Example</h4><br>

    Look at the Emulator.

</form>
<%
```

```

        GraphicPrinter gP = null;

        try{

            Connector conn = Connector.getInstance();

            DriverInfo[] driverInfo =
conn.listDrivers(ConnectionType.PRINTER);

            PrinterParameters params = new
PrinterParameters(driverInfo[0]);

            params.setPrinterMode(PrinterParameters.GRAPHIC_MODE);

            gP = (GraphicPrinter) conn.open(params);

            String[] sFonts =
gP.getFontConfigurationManager().listFontNames();

            PrinterFont pF = gP.getFont(sFonts[0]);





            gP.drawText(pF, 10, 10, "Hello World!", 0);

            gP.doPrint(1);


            gP.close();
        }
        catch (Throwable error){
            error.printStackTrace();
        }
        finally{
            try{
                gP.close();
            }
            catch (Exception ex){
            }
        }
    }
}
%>
</body>
</html>


```

7. Save the file by clicking on  Save.
8. Export the project by pressing the  *MDK: Export a project into a MI archive (.war/.jar)*. Select the project from the *MI Projects* list and hit the *Next* button. Write */initial.jsp* on *URL pattern*.
9. Upload the newly created .war file to the SAP Web Console. Be sure to assign the same name to the application as the name of the .war file itself.
10. Assign the .war file to your MI Client.

11. Start the MI Client and log on to it. Synchronize the MI Client to download the application (.war file). Restart the MI Client.
12. Set on the emulation mode by clicking on  *Set MDK Peripheral Emulation Mode*.
13. Launch the Emulator by clicking on  *Launch the Peripheral I/O Emulator*.
14. Set the emulator Offline by clicking  *On/Offline*, and then click on  *Capability...* In the *General* tab, make sure that *Page Measurements* is not checked. Select the *Media* tab and on *Type* select *Continuous*. When the *Setup Reminder* dialog box comes up, select *Yes*.



If the Setup Reminder dialog box does not come up, click on  *Setup...*



15. In the Printer Setup window, under the *Media* tab, select *Continuous_4* and click *OK*.
16. Run the application by clicking on  *MDK: Export a project into a MI archive (.war/.jar)*. Then click on *Next* and select the checkboxes for *Run the mobile application* and *Uppercase for MI archive name*, in the *URL pattern* field write */initial.jsp*.



To run the application again, choose *Run* → *Run...* Select the project under *Configurations* and click on *Run*. (The *Package Explorer* tab must be selected.)

17. Click on *Finish*. The message "Hello World!" is displayed in the Emulator.

Alternate scenario: printing using the Windows driver:

1. Install the connector and piprmswin32 driver add-ons to the SAP Web Console.
 - **For Windows 2000**, the files are
 - connector_w2k_x86_jvm_1_1_0
 - piprmswin32_w2k_x86_jvm_1_1_0
 - **For Windows XP**, the files are
 - connector_wxp_x86_jvm_1_1_0
 - piprmswin32_wxp_x86_jvm_1_1_0
2. Perform a synchronization to install the Driver Add-Ons.
3. Set the emulation mode off by clicking on  *Set MDK Peripheral Emulation Mode*.
4. Run the application by clicking on  *MDK: Export a project into a MI archive (.war/.jar)*. Then click on *Next* and select the checkboxes for *Run the mobile application* and *Uppercase for MI archive name*, in the *URL pattern* field write */initial.jsp*.
5. Click on *Finish*. "Hello Word" is printed in the default printer configured.



PIOS API Core

Purpose

The **PIOS API Core** includes classes used to discover, connect and use functionality available across different peripheral types. It provides an abstraction layer between the application and the peripheral. A developer using PIOS does not have to be concerned about the implementation details of each peripheral model supported. Instead the developer will target abstracted functionality provided by the peripheral and required by the application.

Implementation Considerations

The abstraction provided by the PIOS API also extends to provide peripheral support across several platforms (OS, JVMs, processors).

Integration

The PIOS API Core is part of the MI Client API.

Features

The PIOS API Core contains common functionality available across peripheral types. For example, it includes methods to:

- discover peripheral drivers installed on the device
- open a connection to a peripheral
- configure the driver



Refer to the MI Client API javadocs for detailed method description.

Constraints

- Peripheral models may offer additional functionality not supported by the PIOS API.



Printer API

Purpose

Printer support in the SAP Mobile Infrastructure is offered as part of the Peripheral Input/Output Services (PIOS). The Printer API provides an abstraction layer between the intricacies of different printer drivers and the application.

Integration

The Printer API is part of the MI Client API.

Features



- Refer to the Printer API javadocs for detailed method description.
- Refer to [Features Description \[Seite 11\]](#).

Constraints

- Printer models may offer additional features not supported by this API.
- Supported functionality varies according to printer make and model.



PIOS Printer API Features Description


Definition

PIOS Printer API provides support for several features. These features vary depending on the printer make and model. The table below provides a description for these features:

Printable Objects

Printable Object	Description
Bitmap Image	Images in bitmap format.
PCX Image	Images in PCX format.
Scalable Fonts	Fonts based on Vector graphics and scale without degradation.
Bitmap Fonts	Fonts made of character images and may present some degradation when scaled up.
Barcode	Barcode printing.

Printer Features

Printer Feature	Description
Clear Error	Clear error condition from the printer.
Advance Forward	Advances the paper forward.
Advance Backward	Retracts the paper.
Dispose	Dispose of the commands sent to the printer. (No information is printed.)
Get Status	Queries the printer status.
Page measurements	Page length and width given in points.
Printer Head width	Obtain the printer head width (in points).
Text metrics	Obtain the text dimension (in points) based on the font.
Barcode metrics	Obtain the barcode dimension (in points).
Image metrics	Obtain the image dimension (in points).
Load Image	Upload an image into the printer's memory.
Delete Image	Delete an image from the printer's memory.
Get DPI (Dot per inch)	Returns the printer's resolution.
Transport Configuration	Modify and set the default transport (i.e. Serial, Bluetooth) options.
Font Configuration	Configure (add, delete, change) font information.
Send Raw Data	<p>Sends raw bytes to the printer. This option sends the bytes (information) submitted directly to the printer, without any intervention from the API.</p> <div style="text-align: center;">  <p>This feature should be used with caution. The printer may exhibit unexpected behavior if invalid data is sent to it.</p> </div>

Graphic Mode Printing

Graphic Mode Feature	Description
Barcode Rotation	Barcode rotation in 90° increments. Barcodes can be rotated to 90°, 180°, and 270°.
Image Rotation	Image rotation in 90° increments. Images can be rotated to 90°, 180°, and 270°.
Text Rotation	Text rotation in 90° increments. Text can be rotated to 90°, 180°, and 270°.
Line thickness	Set the line thickness. (in points)
Draw Text	Print text.
Page measurements	Page length and width given in points (for non-continues paper).
Draw Barcode	Print barcodes.
Draw Line	Print lines.

Draw Rectangle	Print rectangles.
Draw Image	Print images.

Line Mode Printing

Line Mode Feature	Description
Barcode Alignment	Align a barcode in a line (center, right, left).
Image Alignment	Align an image in a line (center, right, left).
Text Alignment	Align text in a line (center, right, left).
Print Text	Print text in a line.
Print Barcode	Print barcodes in a line.
Print Image	Print images in a line.
Set Line Spacing	Defines the space in points between two lines (in points).

Symbology

Symbology	Description
Codabar	Support for Codabar barcode symbology.
Code 39	Support for Code 39 barcode symbology.
Code 128	Support for Code 128 barcode symbology.
EAN-8	Support for EAN-8 barcode symbology.
EAN-13	Support for EAN-13 barcode symbology.
Interleaved 2 of 5	Support for Interleaved 2 of 5 barcode symbology.
PDF417	Support for PDF-417 (Portable Data Format) 2D barcode symbology.
UCC/EAN-128	Support for UCC/EAN barcode symbology.
UPC-A	Support for Universal Product Code (UPC) barcode symbology.



PIOS Printer API Guidelines

Definition

This document explains several guidelines for the PIOS Printer API of the SAP NetWeaver Developer Studio. Each of the guidelines is discussed below.

Use

These guidelines are intended for all the developers working on the SAP NetWeaver Developer Studio in a mobile application with printer peripheral requirements. It helps the developer get the most out of the PIOS infrastructure. The developer can access these features provided by PIOS through the MI Client API.

1. Printer Connection

Since multiple applications can be running in a mobile device at a given time it is recommended that mobile applications should open the connection to the printer when they are ready to send data to the printer and close the connection when they have finished. This prevents other applications from having to wait for the first one to close before they can open a connection. Otherwise if a mobile application opens the printer connection when it starts and does not close it until it finishes, no other application may use the printer until the first one finishes. Other mobile applications waiting to open a connection to the printer may become unresponsive because they are not capable of establishing the connection.

2. DPI Awareness

DPI stands for “Dots Per Inch”, and it is the standard unit of measure used for printer resolution. Different printer models may have different resolutions. A developer should be aware of this fact while writing an application that works with more than one printer model. If the application prints an image, the size of the printed image may vary depending on the printer resolution. For instance, an image printed with a resolution of 300 dpi may be twice as large as an image printed with a 600 dpi resolution. Developers should write applications handling possible DPIs.

3. Printer Head Width

Printer head width information, returned in points, should be used by MI developers to avoid hard-coding this information into mobile applications. Doing so will limit the capabilities to run the application on multiple printer models.

4. Page Size Awareness

If an application is written for a printer that uses the *Page measurements* attribute (only supported for non-continuous paper printers), it can take advantage of this feature and position the fields to be printed according to the paper size. The term “fields” as used in this paragraph refers to text, lines, images, and barcodes.

5. Use of Field Metrics

Given the fact that printers vary in size, developers should make use of the field and page metrics to position images, barcodes, and graphics on the printed page.

6. Image Handling

Since loading images is time consuming, images should be loaded during the application installation cycle. In other words, the loading and deletion of images should not occur continuously throughout the execution of the mobile application.

7. Font Mapping

The API has virtual fonts that map to physical fonts in the printer. Virtual fonts can be added, removed, or modified. It is important to remember that if a font (virtual) is added in the API, then its counterpart, the physical font, must be added to the printer as well. Also a new virtual font can be set to be a copy of an existing font with modified attributes. The font name parameter must exist in the printer.

8. Scalable and Bitmapped Fonts

It is recommended to use scalable fonts over bitmapped fonts whenever possible (not all printers support both font types). Scalable fonts make better use of higher resolutions because they are based on vector graphics, and are more easily scaled than bitmapped. Bitmapped fonts may make their pixels evident when their size is enlarged.

9. Printing Copies

Printing several copies of the same data should be done by providing the desired number of copies to the `doPrint()` method. There is no need to go through another print cycle, sending the same data for a copy.

10. Un-buffered Methods

These are methods that are directly sent to the printer. They do not pass through the printer buffer and therefore cannot be cancelled. The un-buffered methods are `clearError()`, `getStatus()`, and `advance()`.



The advance method must be used only to position the print head to a specific position on the paper. To move forward in line mode developers should use the `printText()` method using an empty string.

11. Raw Bytes

A send raw bytes method exists but its use is not encouraged as it ties applications to specific printer models. The resulting behavior, for other printer models, cannot be predicted.



Printer API Examples

This section contains Printer API examples.



Print Text on the Left Paper

In this example the Printer is in **Line** mode and a line is printed with left alignment. The program performs the following steps:

1. Opens the connection to the printer.
2. Prints the sample text with left alignment.
3. Closes the connection to the printer.

```
package test;

import com.sap.ip.me.api.pios.connection.*;
import com.sap.ip.me.api.pios.printer.*;
public class TextLeftLineMode {

    public static void main(String[] args) {

        LinePrinter lP = null;
        try{
            Connector conn = Connector.getInstance();
            DriverInfo[] driverInfo =
conn.listDrivers(ConnectionType.PRINTER);
            PrinterParameters params = new
PrinterParameters(driverInfo[0]);
            params.setPrinterMode(PrinterParameters.LINE_MODE);

            //----- (1) -----
            lP = (LinePrinter) conn.open(params);

            String[] sFonts =
lP.getFontConfigurationManager().listFontNames();
            PrinterFont pF = lP.getFont(sFonts[0]);
            lP.printText(pF, "", LinePrinter.NO_ALIGNMENT);
            //----- (2) -----
            lP.printText(pF, " Left side of the line.",
LinePrinter.ALIGN_LEFT);
            lP.doPrint(1);
        }
        catch (Throwable error){
            error.printStackTrace();
        } finally {
            //----- (3) -----
            try {
                lP.close();
            } catch (Exception ex) {}
        }
        return;
    }
}
```




Print Text Center Alignment in Line Mode

This example demonstrates text aligned in the center of the line with the printer in **Line** mode. The program does the following:

1. Opens the connection to the printer.
2. Prints the text with center alignment.
3. Closes the connection to the printer.

```
package test;

import com.sap.ip.me.api.pios.connection.*;
import com.sap.ip.me.api.pios.printer.*;

public class TextCenterLineMode {

    public static void main(String[] args) {

        LinePrinter lP = null;

        try{

            Connector conn = Connector.getInstance();
            DriverInfo[] driverInfo =
conn.listDrivers(ConnectionType.PRINTER);
            PrinterParameters params = new
PrinterParameters(driverInfo[0]);
            params.setPrinterMode(PrinterParameters.LINE_MODE);

            //-----(1)-----

            lP = (LinePrinter) conn.open(params);

            String[] sFonts =
lP.getFontConfigurationManager().listFontNames();

            PrinterFont pF = lP.getFont(sFonts[0]);

            lP.printText(pF, "", LinePrinter.NO_ALIGNMENT);

            //-----(2)-----

            lP.printText(pF, "Center of the line.",
LinePrinter.ALIGN_CENTER);
            lP.doPrint(1);
```

```
    }
    catch (Throwable error){
        error.printStackTrace();
    } finally {
        //------(3)-----
        try {
            lP.close();
        } catch (Exception ex) {}
    }
    return;
}
}
```



Print Text on the Right Side

In this example the printer is in **Line** mode and the text is printed on the right side of the page. This program follows the steps below:

1. Opens the connection to the printer.
2. Prints the text with right alignment.
3. Closes the connection to the printer.

```
package test;

import com.sap.ip.me.api.pios.connection.*;
import com.sap.ip.me.api.pios.printer.*;

public class TextRightLineMode {

    public static void main(String[] args) {

        LinePrinter lP = null;

        try{

            Connector conn = Connector.getInstance();
            DriverInfo[] driverInfo =
conn.listDrivers(ConnectionType.PRINTER);
            PrinterParameters params = new
PrinterParameters(driverInfo[0]);
```

```

        params.setPrinterMode(PrinterParameters.LINE_MODE);

        //----- (1) -----

        lP = (LinePrinter) conn.open(params);

        String[] sFonts =
lP.getFontConfigurationManager().listFontNames();
        PrinterFont pF = lP.getFont(sFonts[0]);

        lP.printText(pF, "", LinePrinter.NO_ALIGNMENT);
        //----- (2) -----

        lP.printText(pF, "Right side of the line. ",
LinePrinter.ALIGN_RIGHT);
        lP.doPrint(1);

    }
    catch (Throwable error){
        error.printStackTrace();
    } finally {
        //----- (3) -----

        try {
            lP.close();
        } catch (Exception ex) {}
    }
    return;
}
}

```



Text Drawn on the Left Side of the Page

This example draws the text on the left of the page. The printer is in **Graphic** mode. This program follows the steps below:

1. Opens the connection to the printer.
2. Calculates the bottom of the page using the page and font metrics.
3. Draws the text on the bottom left corner of the page.
4. Closes the connection to the printer.

```
package test;

import com.sap.ip.me.api.pios.connection.*;
import com.sap.ip.me.api.pios.printer.*;

public class TextLeftGraphicMode {
    public static void main(String[] args) {
        float xLeft = 0, yBottom = 0;

        String sText="This text is on the left and bottom of the
page.";

        GraphicPrinter gP = null;
        try{
            Connector conn = Connector.getInstance();
            DriverInfo[] driverInfo =
conn.listDrivers(ConnectionType.PRINTER);
            PrinterParameters params = new
PrinterParameters(driverInfo[0]);
            params.setPrinterMode(PrinterParameters.GRAPHIC_MODE);
            //----- (1) -----
            gP = (GraphicPrinter) conn.open(params);
            String[] sFonts =
gP.getFontConfigurationManager().listFontNames();
            PrinterFont pF = gP.getFont(sFonts[0]);
            //----- (2) -----
            yBottom = gP.getPageMetrics().getHeight() -
pF.getMetrics(sText).getHeight() - 10;
            //----- (3) -----
            gP.drawText(pF, xLeft, yBottom, sText,
GraphicPrinter.NO_ROTATION);
            gP.doPrint(1);
        }
        catch (Throwable error){
            error.printStackTrace();
        } finally {
            //----- (4) -----
            try {
                gP.close();
            } catch (Exception ex) {}
        }
        return;
    }
}
```



Draw Text in the Center of the Page in Graphic Mode

This example demonstrates a line printed in the center of the page with the printer in **Graphic** mode. The program does as explained below:

1. Opens connection to the printer
2. Using the page and font metrics, calculates the coordinates for the text to be in the center of the page.
3. Draws the text on the center of the page.
4. Closes the connection to the printer.

```
package test;

import com.sap.ip.me.api.pios.connection.*;
import com.sap.ip.me.api.pios.printer.*;

public class TextCenterGraphicMode {

    public static void main(String[] args) {
        float xCenter=0, yCenter=0;
        String sText="This text is on the center.";

        GraphicPrinter gP = null;
        try{

            Connector conn = Connector.getInstance();
            DriverInfo[] driverInfo =
conn.listDrivers(ConnectionType.PRINTER);
            PrinterParameters params = new
PrinterParameters(driverInfo[0]);
            params.setPrinterMode(PrinterParameters.GRAPHIC_MODE);

            //----- (1) -----
            gP = (GraphicPrinter) conn.open(params);

            String[] sFonts =
gP.getFontConfigurationManager().listFontNames();

            //----- (2) -----
            PrinterFont pF = gP.getFont(sFonts[0]);
```

```

        yCenter = (gP.getPageMetrics().getHeight()/2) -
        (pF.getMetrics(sText).getHeight()/2);

        Metrics metrics = pF.getMetrics(sText);
        float pageWidth = gP.getPageMetrics().getWidth();
        xCenter = (pageWidth/2) - (metrics.getWidth()/2);

        //----- (3) -----

        gP.drawText(pF, xCenter, yCenter, sText,
        GraphicPrinter.NO_ROTATION);
        gP.doPrint(1);

    }
    catch (Throwable error){
        error.printStackTrace();
    } finally {
        //----- (4) -----
        try {
            gP.close();
        } catch (Exception ex) {}
    }
    return;
}
}

```



Text Drawn on the Right Side of the Page

In this example the code draws text on the right side of the page. The printer is in **Graphic** mode. The program does as follows:

1. Opens the connection to the printer.
2. Uses page and font metrics to calculate the coordinates for the text to print on the right.
3. Draws the text at the calculated coordinates.
4. Closes the connection to the printer.

```

package test;

import com.sap.ip.me.api.pios.connection.*;
import com.sap.ip.me.api.pios.printer.*;

public class TextRightGraphicMode {

```

```
public static void main(String[] args) {  
    float xRight=0, y=15;  
    String sText="This text is on the right.";  
  
    GraphicPrinter gP = null;  
  
    try{  
  
        Connector conn = Connector.getInstance();  
        DriverInfo[] driverInfo =  
conn.listDrivers(ConnectionType.PRINTER);  
        PrinterParameters params = new  
PrinterParameters(driverInfo[0]);  
        params.setPrinterMode(PrinterParameters.GRAPHIC_MODE);  
  
        //----- (1) -----  
        gP = (GraphicPrinter) conn.open(params);  
  
        String[] sFonts =  
gP.getFontConfigurationManager().listFontNames();  
  
        PrinterFont pF = gP.getFont(sFonts[0]);  
        //----- (2) -----  
        xRight = gP.getPageMetrics().getWidth() -  
pF.getMetrics(sText).getWidth() - 10;  
        //----- (3) -----  
        gP.drawText(pF, xRight, y, sText,  
GraphicPrinter.NO_ROTATION);  
        gP.doPrint(1);  
    }  
    catch (Throwable error){  
        error.printStackTrace();  
    } finally {  
        //----- (4) -----  
        try {  
            gP.close();  
        } catch (Exception ex) {}  
    }  
    return;  
}  
}
```



Two Page Report

This small program prints a two page report. The printer is in **Graphic** mode. This program does the following:

1. Opens connection to the printer.
2. Draws the first page with a sample text.
3. Draws the second page with another sample text.
4. Closes the connection to the printer.

```
package test;

import com.sap.ip.me.api.pios.connection.*;
import com.sap.ip.me.api.pios.printer.*;

public class TwoPageReport {

    public static void main(String[] args) {
        int x=15, y=17;

        GraphicPrinter gP = null;

        try{

            Connector conn = Connector.getInstance();
            DriverInfo[] driverInfo =
conn.listDrivers(ConnectionType.PRINTER);
            PrinterParameters params = new
PrinterParameters(driverInfo[0]);
            params.setPrinterMode(PrinterParameters.GRAPHIC_MODE);

            //-----(1)-----

            gP = (GraphicPrinter) conn.open(params);

            String[] sFonts =
gP.getFontConfigurationManager().listFontNames();
            PrinterFont pF = gP.getFont(sFonts[0]);

            //-----(2)-----

            gP.drawText(pF, x, y, "First Page", 0);
            gP.doPrint(1);
```



```
//----- (3) -----  
gP.drawText(pF, x, y, "Second Page", 0);  
gP.doPrint(1);  
  
}  
catch (Throwable error){  
    error.printStackTrace();  
} finally {  
    //----- (4) -----  
    try {  
        gP.close();  
    } catch (Exception ex) {}  
}  
return;  
}  
}
```



Image Printing with Printer in Line Mode

This example demonstrates how to print an image with the printer in **Line** mode. In this case there is a bitmap image file called "test.bmp", which is stored in "C:\TEMP\". This program does as explained below:

1. Lists the drivers.
2. Opens the image file.
3. Opens connection to the printer.
4. Loads the image into the printer.
5. Prints the image.
6. Closes connection to the printer.

```
package test;  
  
import java.io.BufferedInputStream;  
import java.io.File;  
import java.io.FileInputStream;  
import com.sap.ip.me.api.pios.connection.*;  
import com.sap.ip.me.api.pios.printer.*;  
  
public class ImageLineMode {  
  
    public static void main(String[] args) {
```

```
LinePrinter lP = null;
PrinterImage image = null;

try {

    Connector conn = Connector.getInstance();

    //----- (1) -----
    DriverInfo[] driverInfo =
conn.listDrivers(ConnectionType.PRINTER);
    PrinterParameters params = new
PrinterParameters(driverInfo[0]);
    params.setPrinterMode(PrinterParameters.LINE_MODE);

    try {
        //----- (2) -----
        File testImage = new File("C:\\TEMP\\test.bmp");
        if (testImage.exists()) {
            BufferedInputStream bfis = null;

            try {
                bfis = new BufferedInputStream(new
FileInputStream(testImage));

                int bufferSize = (int)testImage.length();
                byte[] imageBytes = new byte[bufferSize];

                int i = bfis.read(imageBytes, 0, bufferSize);

                //----- (3) -----
                lP = (LinePrinter) conn.open(params);

                if (i > 0) {
                    image = lP.createImage("test",
PrinterImage.IMAGE_BMP, imageBytes);

                    //----- (4) -----
                    lP.loadImage(image);
                }
            }
        }
    } finally {
```

```

        if (bfis != null) bfis.close();
    }
}
catch (Throwable tFile) {
    tFile.printStackTrace();
}
String[] sFonts =
lP.getFontConfigurationManager().listFontNames();
PrinterFont pF = lP.getFont(sFonts[0]);

lP.printText(pF, "", LinePrinter.NO_ALIGNMENT);

//----- (5) -----
lP.printImage(image.getName(), LinePrinter.ALIGN_CENTER);
lP.doPrint(1);

}
catch (Throwable error){
    error.printStackTrace();
} finally {
    //----- (6) -----
    try {
        lP.close();
    } catch (Exception ex) {}
}
return;
}
}

```



Printing a Barcode with Printer in Line Mode

In this code a barcode is printed using Code 39 symbology. The "Full ASCII" and "Check Digit mod43" options are set for the symbology. This code performs the following:

1. Opens connection to the printer.
2. Creates a barcode with Human Readable Below.
3. Sets the density for the barcode to the default.
4. Sets the height of the barcode to 10.
5. Sets the Scale of the barcode to Double.
6. Encodes the data used, 10101, using ASCII.

7. Sends barcode to the printer.
8. Closes connection to the printer.

```
package test;

import com.sap.ip.me.api.pios.connection.*;
import com.sap.ip.me.api.pios.printer.*;
import com.sap.ip.me.api.pios.symbology.*;

public class BarcodeLineMode {

    public static void main(String[] args) {

        LinePrinter lP = null;

        try{
            Connector conn = Connector.getInstance();
            DriverInfo[] driverInfo =
conn.listDrivers(ConnectionType.PRINTER);
            PrinterParameters params = new
PrinterParameters(driverInfo[0]);
            params.setPrinterMode(PrinterParameters.LINE_MODE);

            Symbology symbology = new Code39();
            symbology.setOptions(Code39.FULLASCII |
Code39.CHECK_DIGIT_MOD43);

            //----- (1) -----
            lP = (LinePrinter) conn.open(params);
            String[] sFonts =
lP.getFontConfigurationManager().listFontNames();
            PrinterFont pF = lP.getFont(sFonts[0]);

            //----- (2) -----
            PrinterBarcode barcode = lP.createBarcode(symbology,
PrinterBarcode.HUMAN_READABLE_BELOW);

            //----- (3) -----
            barcode.setDensity(PrinterBarcode.DENSITY_DEFAULT);

            //----- (4) -----
            barcode.setHeight(10);
```

```
//----- (5) -----  
barcode.setScaleFactor(PrinterBarcode.SCALE_DOUBLE);  
  
//----- (6) -----  
byte[] data = "10101".getBytes("ASCII");  
  
//----- (7) -----  
lP.printText(pF, "", LinePrinter.NO_ALIGNMENT);  
lP.printBarcode(barcode, data, LinePrinter.ALIGN_CENTER);  
  
lP.doPrint(1);  
  
}  
catch (Throwable error){  
    error.printStackTrace();  
} finally {  
    //----- (8) -----  
    try {  
        lP.close();  
    } catch (Exception ex) {}  
}  
return;  
}  
}
```



Text Rotation

This example demonstrates text rotation. The printer is in **Graphic** mode. This program follows the steps below:

1. Opens the connection to the printer.
2. Gets the text metrics, width and height.
3. Draws the text with a rotation of 180 degrees.
4. Closes the connection to the printer.

```
package test;  
  
import com.sap.ip.me.api.pios.connection.*;  
import com.sap.ip.me.api.pios.printer.*;
```

```
public class RotateText {

    public static void main(String[] args) {
        float x=0, y=0;
        String sText="This is a line of text.";

        GraphicPrinter gP = null;
        try{

            Connector conn = Connector.getInstance();
            DriverInfo[] driverInfo =
conn.listDrivers(ConnectionType.PRINTER);
            PrinterParameters params = new
PrinterParameters(driverInfo[0]);
            params.setPrinterMode(PrinterParameters.GRAPHIC_MODE);

            //-----(1)-----
            gP = (GraphicPrinter) conn.open(params);

            String[] sFonts =
gP.getFontConfigurationManager().listFontNames();

            PrinterFont pF = gP.getFont(sFonts[0]);

            //-----(2)-----
            x = pF.getMetrics(sText).getWidth() + 10;
            y = pF.getHeight() + 15;

            //-----(3)-----
            gP.drawText(pF, x, y, sText,
GraphicPrinter.ROTATE_180_DEGREES);
            gP.doPrint(1);

        }
        catch (Throwable error){
            error.printStackTrace();
        } finally {
            //-----(4)-----
            try {
                gP.close();
            } catch (Exception ex) {}
        }
    }
}
```

```
    }  
    return;  
  }  
}
```



Rotate an Image

This example below shows demonstrates rotating an image using the PIOS Client API. In this case there is a bitmap image file called "test.bmp", which is stored in "C:\TEMP\". The printer is **Graphic** mode. This program follows the steps below:

1. Opens the image file.
2. Opens the connection to the printer.
3. Loads the image to the printer's memory.
4. Gets the image height.
5. Draws the image with a 90 degree rotation.
6. Closes the connection to the printer.

```
package test;  
  
import java.io.BufferedReader;  
import java.io.File;  
import java.io.FileInputStream;  
import com.sap.ip.me.api.pios.connection.*;  
import com.sap.ip.me.api.pios.printer.*;  
  
public class RotateImage {  
  
    public static void main(String[] args) {  
        GraphicPrinter gP = null;  
        PrinterImage image = null;  
        float x = 0, y = 15;  
  
        try {  
  
            Connector conn = Connector.getInstance();  
            DriverInfo[] driverInfo =  
conn.listDrivers(ConnectionType.PRINTER);  
            PrinterParameters params = new  
PrinterParameters(driverInfo[0]);  
            params.setPrinterMode(PrinterParameters.GRAPHIC_MODE);
```

```
try {
    //-----(1)-----
    File testImage = new File("C:\\TEMP\\test.bmp");
    if (testImage.exists()) {
        BufferedInputStream bfis = null;

        try {
            bfis = new BufferedInputStream(new
FileInputStream(testImage));

            int bufferSize = (int)testImage.length();
            byte[] imageBytes = new byte[bufferSize];

            int i = bfis.read(imageBytes, 0, bufferSize);

            //-----(2)-----
            gP = (GraphicPrinter) conn.open(params);

            if (i > 0) {
                image = gP.createImage("test",
PrinterImage.IMAGE_BMP, imageBytes);

                //-----(3)-----
                gP.loadImage(image);
            }
        }
        finally {
            if (bfis != null) bfis.close();
        }
    }
} catch (Throwable tFile) {
    tFile.printStackTrace();
}

//-----(4)-----
x = image.getMetrics().getHeight() + 10;

//-----(5)-----
gP.drawImage(image.getName(), x, y,
GraphicPrinter.ROTATE_90_DEGREES);
```



```

        gP.doPrint(1);

    }
    catch (Throwable e){
        e.printStackTrace();
    } finally {
        //----- (6) -----
        try {
            gP.close();
        } catch (Exception ex) {}
    }
    return;
}
}

```



Rotating a Barcode

This example demonstrates how to rotate a barcode. Rotation of a barcode is only possible with the printer in **Graphic** mode. This program does the following:

1. Sets the symbology options for Full ASCII and Check Digit Mod43.
2. Opens the connection to the printer.
3. Sets for the barcode: Human Readable Below, Default Density, Double Scale.
4. Encodes the data, 32123, using ASCII.
5. Gets the metrics for the barcode.
6. Draws the barcode rotating it 270 degrees.
7. Closes the connection to the printer.

```

package test;

import com.sap.ip.me.api.pios.connection.*;
import com.sap.ip.me.api.pios.printer.*;
import com.sap.ip.me.api.pios.symbology.*;

public class RotateBarcode {

    public static void main(String[] args) {
        float x=0, y=0;

        GraphicPrinter gP = null;
        try{

```

```

        Connector conn = Connector.getInstance();

        DriverInfo[] driverInfo =
conn.listDrivers(ConnectionType.PRINTER);

        PrinterParameters params = new
PrinterParameters(driverInfo[0]);

        params.setPrinterMode(PrinterParameters.GRAPHIC_MODE);

        Symbology symbology = new Code39();

        //----- (1) -----
        symbology.setOptions(Code39.FULLASCII |
Code39.CHECK_DIGIT_MOD43);

        //----- (2) -----
        gP = (GraphicPrinter) conn.open(params);

        //----- (3) -----
        PrinterBarcode barcode = gP.createBarcode(symbology,
PrinterBarcode.HUMAN_READABLE_BELOW);
        barcode.setDensity(PrinterBarcode.DENSITY_DEFAULT);
        barcode.setHeight(10);
        barcode.setScaleFactor(PrinterBarcode.SCALE_DOUBLE);

        //----- (4) -----
        byte[] data = "32123".getBytes("ASCII");

        //----- (5) -----
        x = barcode.getHeight() + 10;
        y = barcode.getMetrics(data).getWidth() + 15;

        //----- (6) -----
        gP.drawBarcode(barcode, x, y,
data, GraphicPrinter.ROTATE_270_DEGREES);
        gP.doPrint(1);

    }
    catch (Throwable error){
        error.printStackTrace();
    } finally {
        //----- (7) -----
        try {

```

```

        gP.close();
    } catch (Exception ex) {}
}
return;
}
}

```



Configure the Printer Parameters

This example demonstrates how to configure a printer parameter in the piprmswin32 driver. The parameter to display a printer selection pop-up window is set to true, used, and then set to false. The printer is in **Line** mode. This program follows the steps mentioned below:

1. Opens connection to the printer.
2. Gets the configuration manager for the driver.
3. Using the configuration manager gets the parameter of the driver.
4. Sets the value of the parameter to true.
5. Closes and reopens connection to the printer for the change to the parameter value to take effect.
6. Prints some sample text.
7. Finally, sets the parameter back to false.
8. Closes the connection to the printer.

```

package test;

import com.sap.ip.me.api.pios.connection.*;
import com.sap.ip.me.api.pios.printer.*;
import com.sap.ip.me.api.pios.configuration.*;

public class ConfigurePrinterParameters {

    public static void main(String[] args) {

        LinePrinter lP = null;
        try {
            //-----(1)-----
            Connector conn = Connector.getInstance();
            DriverInfo[] driverInfo =
conn.listDrivers(ConnectionType.PRINTER);
            PrinterParameters params = new
PrinterParameters(driverInfo[0]);

```

```
params.setPrinterMode(PrinterParameters.LINE_MODE);

lP = (LinePrinter) conn.open(params);

//----- (2) -----
DriverConfigurationManager manager =
lP.getParameters().getDriver().getConfigurationManager();

//----- (3) -----
Configuration config1 =
manager.getConfiguration("DriverParameters");

//----- (4) -----
String promptUser = config1.getParameters()[0];
config1.setParameterValue(promptUser, "true");
manager.save();

//----- (5) -----
lP.close();
lP = (LinePrinter) conn.open(params);

//----- (6) -----
String[] sFonts =
lP.getFontConfigurationManager().listFontNames();
PrinterFont pF = null;
if (sFonts != null) {
    pF = lP.getFont(sFonts[0]);
    lP.printText(pF, "", LinePrinter.NO_ALIGNMENT);
    lP.printText(pF, " This is a test string.",
LinePrinter.NO_ALIGNMENT);
    pF = lP.getFont(sFonts[1]);
    lP.printText(pF, "", LinePrinter.NO_ALIGNMENT);
    lP.printText(pF, " This is a test string.",
LinePrinter.NO_ALIGNMENT);

    lP.doPrint(1);
}

//----- (7) -----
manager =
lP.getParameters().getDriver().getConfigurationManager();
config1 = manager.getConfiguration("DriverParameters");
```

```
        promptUser = config1.getParameters()[0];
        config1.setParameterValue(promptUser, "false");
        manager.save();

    }
    catch (Throwable error){
        error.printStackTrace();
    }
    finally {
        //----- (8) -----
        try {
            lP.close();
        } catch (Exception ex) {}
    }

    return;
}
}
```



Adding/Remove a Font Using the Client API

This example shows how to add and remove a font configuration to the PIOS infrastructure using the Client API. In this example the “Batang” font configuration is added, some text is printed with this font configuration, and then it is deleted. The font configuration added must map to an existing font in the printer. The printer is in **Line** mode. This code does is as described below:

1. Opens connection to the printer.
2. Adds the font configuration.
3. Sets the parameters for the font configuration and saves the changes.
4. Closes and reopens the connection to the printer for the font configuration to take effect.
5. Prints a sample text.
6. Deletes the font configuration and saves the changes.
7. Closes the connection to the printer.



Remember to configure the [Emulator \[Seite 56\]](#) or printer accordingly.

```
package test;

import com.sap.ip.me.api.pios.connection.*;
```

```
import com.sap.ip.me.api.pios.printer.*;
import com.sap.ip.me.api.pios.configuration.*;

public class AddFont {

    public static void main(String[] args) {

        LinePrinter lP = null;
        try{
            Connector conn = Connector.getInstance();
            DriverInfo[] driverInfo =
conn.listDrivers(ConnectionType.PRINTER);
            PrinterParameters params = new
PrinterParameters(driverInfo[0]);
            params.setPrinterMode(PrinterParameters.LINE_MODE);

            //-----(1)-----
            lP = (LinePrinter) conn.open(params);

            //-----(2)-----
            String sFont = "Batang";
            FontConfigurationManager fCM =
lP.getFontConfigurationManager();
            fCM.addFontConfiguration(sFont);
            Configuration conf = fCM.getFontConfiguration(sFont);

            //-----(3)-----
            conf.setParameterValue("Size", "14");
            conf.setParameterValue("Options", "0");
            conf.setParameterValue("Name", "Batang");
            conf.setParameterValue("Description", "customfont");
            conf.setParameterValue("FontType", "2");
            fCM.save();

            //-----(4)-----
            lP.close();
            lP = (LinePrinter) conn.open(params);

            String[] sFonts =
lP.getFontConfigurationManager().listFontNames();
```

```

//----- (5) -----
PrinterFont pF = lP.getFont(sFont);
lP.printText(pF, "", LinePrinter.NO_ALIGNMENT);
lP.printText(pF, " This is a Batang test string.",
LinePrinter.NO_ALIGNMENT);
pF = lP.getFont(sFonts[1]);
lP.printText(pF, " This is a regular test string.",
LinePrinter.NO_ALIGNMENT);
lP.doPrint(1);

//----- (6) -----
fCM.deleteFontConfiguration(sFont);
fCM.save();

} catch (Throwable error){
    error.printStackTrace();
} finally {
    //----- (7) -----
    try {
        lP.close();
    } catch (Exception ex) {}
}
return;
}
}

```

PIOS Add-on Drivers

The following chapter provides information about the PIOS add-on drivers.



For more information about individual drivers refer to SAP Note 761833.

Installing a Driver Add-on

Purpose

This process is intended for system administrators deploying a mobile application with peripheral requirements. The system administrator deploys drivers and connector add-ons that meet the requirements of the mobile application.

Prerequisites

There is a connector add-on (it can be deployed when the driver is deployed).

Process Flow

1. A mobile application with peripheral requirements is uploaded to the SAP MI Web Console.
2. The system administrator uses the [driver selection tool \[Seite 69\]](#) to find the driver add-on that matches the target platform.
3. The administrator checks if the connector add-on for the target platform is uploaded to the SAP MI Web Console. If the matching driver and/or connector are not loaded to the SAP MI Web Console, they can be obtained from the *SAP Service Marketplace*.
4. After the connector and driver add-ons have been uploaded to the SAP MI Web Console, the administrator deploys them.

If the connector is not installed on the target device, both the connector and the driver add-ons can be deployed at the same time as the application. If there is a connector in the target mobile device, only the driver and application need to be deployed.

Result

The mobile application and the required driver add-on are deployed to the target mobile device.



Driver Configuration

Purpose

Peripheral input output services (PIOS) drivers can be configured by modifying several parameters. Parameters are defined for different configurations, and configurations are divided into configuration types. The system administrator can modify configuration parameters with the Mobile Infrastructure configuration system.

Implementation Considerations

Parameters of driver add-ons are formed with four tokens. The first token is the *driver name*. The second token is the *configuration type*. The third and fourth tokens are *configuration name* and *parameter* respectively. Modification of driver parameters should follow the naming convention presented below:

`<driver name>.<configuration type>.<configuration name>.<parameter>=value`



Tokens and values are case-sensitive. The correct name must be entered in the MI configuration system to change a parameter value.

Integration

- Driver configurations are handled by the MI configuration system.
- Drivers and driver configurations are assigned using the SAP MI Web Console.

Features

Drivers have several parameters that can be used to change options for a driver. These parameters are defined for configurations that are separated into configuration types. PIOS drivers use the driver configuration type (cfg) to store parameters that modify how drivers connect to peripherals. This configuration type is also used to store parameters that are specific from driver to driver. A different configuration type is used to store the font configuration parameters (fntcfg) for peripheral type "Printer". This configuration type (fntcfg) is used to configure fonts supported by the driver.

System administrators can modify driver configuration and font configuration parameters with the MI configuration system.

Constraints

- Adding a font with the font configuration parameters does not install the font on the physical printer. Printer fonts must be installed manually on the printer and should match the configured parameters.
- Configuration values are applied to drivers without validation. Unexpected behavior may be detected if a driver is not configured properly.

Example

Examples for parameter configuration and font configuration parameters are given below:

- In this example, a driver configuration parameter for the piprsymm4t (Symbol microFlash 4t) printer driver is configured. This line sets the serial port baud rate to 9600 bits per second:
`piprsymm4t.cfg.Serial.BaudRate=9600`
- This example modifies a font configuration parameter for the piprmswin32 (Microsoft Windows 32-bit) printer driver. This line sets the "bitmapped bold italic" font options to bold and italic:
`piprmswin32.fntcfg.BitmappedBoldItalic.Options=bold,italic`

See Also

You can find a list of all available parameters in a SAP note that is created for each driver. For a list of available drivers see the collective SAP Note **761833**.



MDK Peripheral Support Actions

Purpose

The MDK Peripheral Support Actions is part of the SAP Mobile Infrastructure perspective of the SAP NetWeaver Developer Studio. It enables the developer to:

1. Create/Modify Driver Requirements Document (DRD)
2. Set MDK Peripheral Emulation Mode
3. Launch the Peripheral I/O Emulator.

Implementation Considerations

The DRD is required for those mobile applications that require some peripheral support. It identifies the peripheral types as well as the attributes within those peripheral types required by an MI application. It is used by both the developer and the system administrator. The developer specifies the peripheral requirements. The system administrator later uses it in the Driver Selection Tool (DST) of the Web Console to find drivers that match those peripheral requirements.

An option to set the MDK in peripheral emulation mode is also provided and enables the developer to automatically switch the output destination from the peripheral to the emulator and vice versa. Launching the Peripheral Input/Output Emulator from SAP Mobile Infrastructure perspective is also possible.

Integration

The MDK Peripheral Support Actions is part of the Mobile Infrastructure perspective of the SAP NetWeaver Developer Studio.

Features

The MDK Peripheral Support Actions has functions to:

- Add a new DRD to an SAP MI project.
 - Add/Remove peripheral types from an existing DRD.
 - Select the required attributes for each peripheral type.
- Set MDK Peripheral Emulation Mode.
- Launch the Peripheral I/O Emulator.



Driver Requirements Document Editor

Definition

The Driver Requirements Document Editor allows the developer to select the peripheral types and their respective options required for the mobile application.

Use

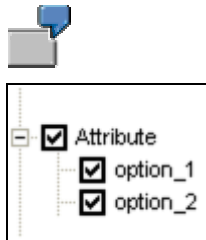
The Driver Requirements Document Editor is displayed:

- After adding a new DRD to an MI project.
- By double clicking an existing DRD in the project.
- After modifying the current DRD.
-

Structure

The DRD Editor has three components:

1. The **attributes tree window**. Displays all available attributes of a peripheral type. It is used to select the required attributes.
 - **Attribute:** Refers to the parent node. When checked, it implies support for all corresponding options under the parent node.
 - **Options:** Refers to each node under an attribute.



Attributes and options available are those supported by the [Printer API \[Seite 11\]](#).

2. The **Description Panel:** displays a short description of the selected attribute. It is the panel located at the bottom of the Driver Requirements Document Editor.
3. **Peripheral Type tab:** Used to select the peripheral attribute tree window from the available peripheral types.



Using the Driver Requirements Document Editor

Use

The editor presents the developer with the attributes tree window. It describes the process of selecting the attributes and options. Is important to select all peripheral attributes and options required by the application in order for the Driver Selection Tool to accurately select peripheral models that match those requirements. Attributes and options that are not selected may or may not be supported by the selected peripheral model.

Features

Two scenarios are supported to select attributes and options in the attributes tree:

1. Select only the parent - When checked, all corresponding attributes options under the root parent node are also checked.
2. Select one or more options - It means required support for all the selected attributes options under the root parent node. They follow a logical AND behavior, each selected option becomes a requirement that must be met during the peripheral add-on selection process.



Printer head width in the Printer peripheral type is the exception to this rule. In this case, when more than one option is selected at least one of them will be matched. In other words, the *Printer head width* attribute follows a logical OR behavior, where the requirement is met when at least one of the selected options is met.

Activities

The developer must check every peripheral option required by the mobile application. Any option that has not been selected is ignored by the Driver Selection Tool. In other words, peripheral drivers displayed in the Driver Selection Tool as matched drivers, may or may not provide support for unchecked options.



MDK Peripheral Support Actions Toolbar

Use

The MDK Peripheral Support Actions Toolbar provides quick access to the Peripheral Support Actions available in the Mobile Infrastructure perspective:






Each of the buttons is further explained below.

Prerequisites

- The SAP NetWeaver Developer Studio is installed.
- An Eclipse project is available and selected.
- The Mobile Infrastructure perspective is selected.

Features

Toolbar Functions

Button	Function Name	Meaning
	<i>Create/Modify Driver Requirements Document</i>	Opens the wizard to create a new Driver Requirements Document. If a DRD is present in the project, opens the wizard to modify it.
	<i>Set MDK Peripheral Emulation Mode</i>	Toggles the Emulation Mode On/Off. Configures the MI Client to use the Peripheral Input / Output emulator instead of the physical peripheral.
	<i>Launch Peripheral I/O Emulator</i>	Launches the Peripheral Input / Output Emulator.



Creating a Driver Requirements Document



Use

This procedure describes the steps required to create a Driver Requirements Document.

Prerequisites

- Eclipse must be in the Mobile Infrastructure perspective.
- An existing project has been selected and the project does not have an existing Driver Requirements Document.

Procedure

3. Click on the *Create/Modify Driver Requirements Document*  button. The *New Driver Requirements Document* wizard opens.
4. 
5. You can also access the wizard by choosing *New* → *Other...* → *MDK Development Tools* on the *File* menu and selecting *Driver Requirements Document*.
6. Select the peripherals required by the mobile application from the *Peripherals Available* list. Click *Add >*.
7. After all peripheral types have been selected, click *Finish*. The *Driver Requirements Document Editor* appears as part of the selected project.

Result

A Driver Requirements Document is created.



Modifying the Driver Requirements Document



Use

Modify an existing Driver Requirements Document (DRD). It allows additions or deletions of peripheral types from an existing DRD.

Prerequisites

- The Mobile Infrastructure perspective is selected.
- A project containing a Driver Requirements Document is selected.

Procedure

1. Click on the *Create/Modify Driver Requirements Document*  button. The *Modify existing Driver Requirements Document* wizard opens.
2. 
3. You can also access the wizard by choosing *New* → *Other...* → *MDK Development Tools* on the *File* menu and selecting the Driver Requirements Document. If a DRD is present in the project, the *Modify existing Driver Requirements Document* wizard will open.
4. Add or remove peripheral types as desired.
5. Click Finish. The *Driver Requirements Document Editor* opens.

Result

The Driver Requirements Document is modified.



Launching the Peripheral I/O Emulator

Use

Launch the Peripheral Input/Output Emulator from the Mobile Infrastructure perspective.

Prerequisites

- Eclipse must be in the Mobile Infrastructure perspective.

Procedure

1. Click on the *Launch the Peripheral I/O Emulator*  button. The emulator is launched.



You can also launch the emulator by selecting *Launch the Peripheral I/O Emulator* on the *Edit* menu.



Refer to the [Peripheral Input/Output Emulator \[Seite 48\]](#) for further details on the use of the emulator.

Result

The emulator is launched.



Set MDK Peripheral Emulation Mode

Use

This mode enables the developer to automatically switch use of the peripheral to the emulator and vice versa. When the developer turns on the Emulation Mode, the application, while running inside the MI Client, uses peripheral emulator.

Prerequisites

- The SAP NetWeaver Developer Studio is in the Mobile Infrastructure perspective.

Procedure

1. Turn on the emulation mode by clicking on the *Set MDK Peripheral Emulation Mode*



button.



You can also set the MDK in emulation mode by selecting *Set MDK Peripheral Emulation Mode* from the *Edit* menu.

2. Restart the MI Client for the settings to take effect.
3. Run your application.

Result

Your application will use the Peripheral I/O emulator.



Peripheral Input/Output Emulator

Purpose

The Peripheral Input/Output Emulator emulates peripheral types supported by PIOS. It helps a developer to test and/or debug a mobile application that requires peripheral support without using the physical peripheral.

Implementation Considerations

- You can test your application without having to connect any peripheral to the system you are working on.
- The Emulator is supported in the Windows XP and 2000 platforms.

Integration

The Peripheral I/O Emulator is part of the Mobile Infrastructure perspective of the SAP NetWeaver Developer Studio.

Components

The emulator main window has several areas:

- **Hierarchical menus** - Group related program functions within each menu category. The categories are:
 - File - Related to project's data. It saves, opens, and creates a new project. The *New Project*, *Open Project*, *Save Project* and *Save Project As...* file menu options are not available at this time.
 - Edit - This menu option is peripheral specific. Refer to the appropriate peripheral type for the menu description.
 - View - Sets the focus of the application to the project tree or peripheral panel.
 - Peripheral Options - This menu option is peripheral specific. Refer to the appropriate peripheral type for the toolbar description.
 - Help - Displays version information of the Emulator.
- **Toolbars** - Each toolbar provides access to frequently used functions in the menus. The toolbars are:
 - Project Toolbar - Provides shortcuts to functions inside the File menu.
 - Peripheral Data Toolbar - Provides shortcuts to functions in the Edit menu. (It is disabled for the printer.)
 - Peripheral Toolbar - This toolbar is peripheral specific. Refer to the appropriate peripheral type for the menu description.
- **Project Tree** - Contains an entry for each installed Peripheral Type. It is located on the left side of the screen, below the toolbars.
- **Peripheral Panel** - Information on this panel depends on the selected peripheral type. It is located on the right side of the screen, to the right of the Project Tree.

Accelerators have been provided for some menu items. Accelerators are key combinations (Ctrl + AnyKey) which substitute a mouse command.

Each selectable menu item and prompt has a mnemonic. These mnemonics are shown by pressing the ALT key.



If you are using Microsoft Windows XP or Windows 2000 you can enable the mnemonics by going to Control Panel - Display Settings and un-checking, in Windows XP, "Hide underlined letters for keyboard navigation until I press the Alt key" or, in Windows 2000, "Hide keyboard navigation indicators until I use the Alt key".

Constraints

- The Emulator restricts emulated functions to those supported by the PIOS Client API. Actual peripheral hardware may have more functions not included in the Emulator.
- Settings in the Emulator are stored by user. If a developer makes changes in the Emulator for a project, the changes are in effect the next time the Emulator runs, even if it is in a different project.
- Image sizes vary according to the printer resolution. As the resolution goes higher the image size reduces.



PIOS Emulator Menu Options

This section describes the PIOS Emulator Menu options.

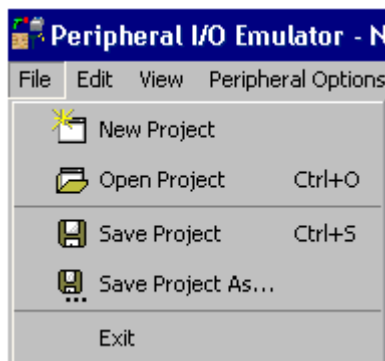


File Menu

Definition

The File menu is used to create, open, and save a project. It also has the *Exit* function to close the Emulator.

The *New Project*, *Open Project*, *Save Project* and *Save Project As...* file menu options are not available at this time.



The Project Toolbar also provides access to these functions:





Edit Menu

Definition

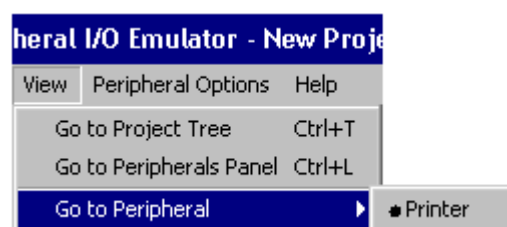
The Edit menu option is not available at this time.



View Menu

Definition

The view menu allows setting the focus to the different panels of the Peripheral I/O Emulator. The focus can be changed quickly to the project tree, the peripheral panel, or you can select a different peripheral type from the available peripheral types, thus changing the peripheral panel and options.



Use

View Menu functions

Function	Accelerator	Description
Go to Project Tree	CTRL + T	Sets the focus on the Project Tree.
Go to Peripherals Panel	CTRL + L	Sets the focus on the Peripherals Panel.
Go to Peripheral		Sets the focus on the Peripherals Panel and changes the current peripheral to the selected one.



Peripheral Options Menu

Definition

The Peripheral Options menu depends on the selected peripheral type.



Refer to [Peripheral Options Menu for Printer Peripheral Type \[Seite 52\]](#).



Help Menu

Definition

The Help menu provides version information of the Emulator.



Use

Help Menu functions

Function	Description
About	Displays version information about the Emulator and available peripheral types.



Printer Peripheral Panel

Definition

The printer peripheral panel contains the virtual printout area, printer buffer, image buffer and the state buffer. These are further discussed below.

Structure

- **Virtual Printout Area** - Emulates the virtual media. It displays images, text, and/or barcodes that are printed. This area is located on the left side of the Print Buffer, the Image Buffer, and the State Buffer.
 - **Measuring String** - used to measure the distance between two points inside the Virtual Printout Area. (Refer to *Using the Measuring String* for details)
 - **Unit button** - used to toggle the reference rulers found on the top and left-hand side of the virtual printout area from inches, points and centimeters. It is located at the top left corner of the printer peripheral panel.



- Any barcodes displayed in the virtual printout are for visual representation purposes only and are not valid barcode representation. Only barcode dimensions and position are emulated
 - Barcode size and font size are an approximation.
 - Images are displayed as black boxes which approximate image size.
- **Print buffer** - Displays the queue of commands sent to the printer. It also keeps a history of commands already printed. This buffer is located at the top right corner of the printer peripheral panel.

- **Image buffer** - Displays images loaded in the emulated image buffer. Located below the Print Buffer.
- **State Buffer** - Displays current emulated printer settings and status: Located below the Image Buffer.
 - Status - Online or Offline.
 - Connection - Open or Closed.
 - Mode - Line or Graphic
 - Resolution - Current emulated resolution in dot per inch (DPI).
 - Print Head - Displays the current position of the printer head.
 - Line Space - Displays the space between lines in points. This field is only relevant in Line Mode.
 - Paper Count - Displays the paper count for non-continuous media. For continuous media this field is blank.
 - Paper Length - Displays the paper length for continuous media. For non-continuous media this field is blank.
 - Media - Displays the media type in use by the emulator.
 - Default Font - Displays the default font type, and size.



Emulator Configuration

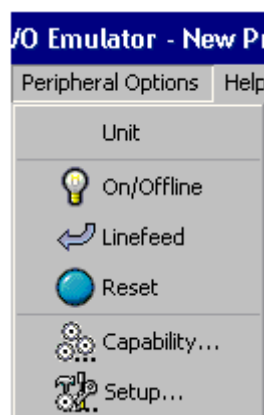
This section describes the PIOS Emulator configuration options.



Peripheral Options Menu for Printer Peripheral Type

Definition

This menu provides access to functions that set and modify the printer emulation behavior. Each function is explained below.










There is also a toolbar, the Peripheral Options Toolbar, which relates directly to this menu:



The relationship between each button and the menu item is further explained in the next table.

Use

Peripheral Options menu functions for the printer

Menu option	Button	Description
Unit		Toggles the ruler unit of measure in the peripheral panel from points to centimeters, to inches, to points again.
On/Offline		Toggles the printer status between Online and Offline.
Linefeed		Moves the printer cursor forward by one line (10 points). Applies to continuous media only.
Reset		Resets the printer. Clears the printer buffer, image buffer, and virtual printout area. Sets the <i>Print Head</i> property value in the <i>State Buffer</i> to (0.0,0.0).
Capability...		<p>This function is only available while the printer is Offline. It provides access to printer configuration attributes in each of the available categories. These categories are explained in the structure section below.</p> <p></p> <p>The PIOS Emulator emulates attributes supported by the Printer API [Seite 11].</p>
Setup...		This function is only available while the printer is Offline. It allows set up of the media, font, and printer resolution.

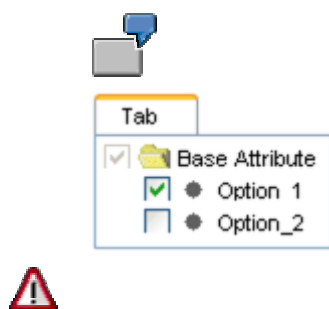
Structure

When you click on *Capability...* a window with several tabs is displayed. Each tab is explained in the next table:

Capability... Categories

Category	Description
General	Contains general printer attributes. The state of each attribute can be modified by clicking on the attribute node selection checkbox or selecting the node and pressing the space bar.
Barcode	Contains a list of the printer barcodes capabilities and barcode attributes. The state of each attribute can be modified by either clicking on the attribute node selection checkbox or selecting the node and pressing the space bar.
Font	Displays the available fonts for emulation. Emulation support for bitmapped, scalable fonts, or both can be selected.
Media	Displays the list of supported paper media. Selection of the type of media, <i>Continuous</i> or <i>Non-Continuous</i> , and the <i>Print Head Width</i> is possible. This automatically selects the media that match these two settings.

- Some attributes displayed in the *General* and *Barcode* tabs are grouped in folders. The first entry inside these folders is referred to as the **base attribute**. This base attribute is automatically selected if any of the other options within the folder is selected. If the base attribute is unselected any other selected entries within the folder are cleared as well.



Under the *Barcode* tab, the Code 39 symbology allows to uncheck *Standard*. However, this is not a real case because whenever a printer supports Code 39, the Standard codeset is supported.

- Selecting a folder will select all attributes in that folder.

Setup... Options

When you click on Setup... in the peripheral options menu or toolbar a window with several tabs is displayed, each tab is explained below:

Option	Description
Media	<p>Select the media (paper size) that the virtual printer will use. Only the ones selected in the Capability Media category will be available for selection in this window.</p> <p>Paper Length / Count - For continuous media, the paper length can be specified in either inches or centimeters.</p> <p>For non-continuous media, the paper count can be specified.</p> <p>Size - Displays the size (width and length) of the media in inches. Note: For continuous paper, the length is represented as zero length</p> <p>Margin - Displays the top, left, bottom and right margins in inches.</p> <p>Pre-print - Displays the width, length, horizontal, and vertical separation of pre-printed media. This is often used to represent labels on a paper backing.</p> <p><i>Margin</i> and <i>Pre-print</i> fields are only showed when there are pre-defined areas in the media.</p>
Default Font	<p>Select default font type, style, and size for the virtual printer.</p> <p>Type - Displays the type (Bitmapped, Scalable) of the default font selected.</p> <p>Style - Select the default font style (PLAIN, BOLD, ITALIC, BOLD+ITALIC)</p> <p>Size - Select the default font size from the list of available sizes.</p>
Resolution	Select the desired resolution for the virtual printer from the list.

Integration

The peripheral options menu contains options specific to an individual peripheral type. Menu options will vary per peripheral type.



Property Files for Printer Peripheral Type

Purpose

The emulator property files offer the option to add or remove available fonts and media from the emulator.

Each property file may have two versions, a default and a user specific version. The default file has a DFT extension, for example "*filename.dft*". The second file has an extension of ".username", that is "*filename.username*". The extension "username" refers to the logon name of the user.

The default versions are created the first time the Emulator runs. The user specific versions are created when the property files are modified through the *Capability...* and/or *Setup...* windows.

Features

All property files follow the Java Property file format:

- The line starting with "#" is a comment line.
- Other lines follow the "Key=Value" format.



If a Key has a space as part of the name, for example "Courier New", a backslash (\) must be inserted before the space. The Key in the file will look like:

```
Courier\ New
```

- When a list is assigned to a Key the values are separated by a coma and no spaces are allowed between the coma and the value. For example, Key1=value1,value2,value3, and so on.

The property files are located in:

ECLIPSE_HOME\plugins\com.sap.ip.me.mdk.pios.docgen_X.Y.Z\emulator\inst\Printer

Where ECLIPSE_HOME refers to the location in the hard disk where Eclipse is installed and X.Y.Z refers to the current version, for example 1.1.0.

Constraints

- The system does not validate for appropriate parameters values. It is strongly recommended to make a backup before modifying a property file.
- Invalid key values may cause unexpected behavior in the emulator. For example, if in *InstalledFontFamilies* a font is added with a space between the coma and the font name, that is ", font", the emulator will start the JVM but no window is displayed.
- The Emulator loads the configuration files when it is started. The Emulator must be restarted whenever a file is modified for the changes to take effect. Otherwise, changes to the files will take effect the next time the Emulator is started.

- Fonts and their respective styles and sizes must be available (installed) in the OS.
- An MI application can only use an installed font if its configuration can be found in the font configuration file. A font can be installed in the Emulator only if it is available in the operating system and its added to the *InstalledFont* file. The font configuration must be added separately in the *pipremulator.fntcfg* file to be available for use from the mobile application.



Installing a Font in the Emulator Printer

Use

To add a font to the available fonts you should modify the file named *InstalledFont.username*.

This file stores the following information:

- Font capabilities of the emulated printer.
- Installed fonts and their details.
- The default font of the emulated printer - This font is only used for displaying the barcode human readable.

The structure of the file is as follows:

Parameter	Description
DefaultFont.Size	The size of the default font. It is specified in unit of measure "point". It must be listed in the available sizes of the font.
bitmappedFont	Supports bitmapped fonts: 1 - yes 0 - no
DefaultFont.Family	The font family name of the default font. It must be in the list of installed font families. Note: it also must conform to the font types that the emulated printer supports, bitmapped or scalable.
scalableFont	Supports scalable fonts: 1 - yes 0 - no
InstalledFontFamilies	The list of installed font family names. The name must be a valid font family name installed in the Operating System where the Printer Emulator is running.
DefaultFont.Style	The style of the default font. It must be included in the available styles of the font.

Font Details: (for a font named *fontname*)

Key	Description
<i>fontname</i> .Sizes	A list of available sizes of the font family, with values separated by comma. The unit of measure for each size is "point".
<i>fontname</i> .Styles	A list of available styles of the font family separated by comma. Possible values are:

	0 : plain 1 : bold 2 : <i>italic</i> 3 : <i>bold and italic</i>
<i>fontname.Type</i>	Font type possible values are: 1: bitmapped font type 2: scalable font type



Adding a font to this file is the equivalent of loading a font into the printer's memory (in the physical printer). The client API will not be able to use the added font unless it is added to the available font configurations. (Refer to [Installing a Font Configuration \[Seite 64\]](#) for more information.)

Prerequisites

- The new font must be a valid font installed in the Operating System on which the Printer Emulator is running.

Procedure



Read [Property Files for Printer Peripheral Type \[Seite 55\]](#).

1. Make a backup copy of the file named *InstalledFont.username*, where username refers to the logon name of the user.



The developer can also modify the default configuration by editing the file named *InstalledFont.dft*. For the changes to take effect in the developer's emulator, delete the file named *InstalledFont.username*. The next time the Emulator opens, it will be configured based on the default file.



Changes made to the default file affect any user who launches the Emulator for the first time after these changes are done.

2. Open the file named *InstalledFont.username* in a text editor.
3. Add the font family name to list of supported font family names. (Refer to before and after example below.)
4. Add all three font detail lines: *Sizes*, *Styles*, and *Type*. (Refer to before and after example below.)

Result

The new font is installed in the Emulator.

Example

The following is an example of the file. The sequence of the lines has been adjusted and blank lines have been added for ease of reading.

```
#
#Wed Jun 23 11:29:33 EDT 2004
bitmappedFont=1
scalableFont=1

InstalledFontFamilies=SansSerif,Arial

SansSerif.Sizes=8.0,10.0,12.0,14.0,16.0,18.0,20.0,22.0,24.0
SansSerif.Styles=0,1,2,3
SansSerif.Type=2

Arial.Sizes=8.0,12.0,16.0
Arial.Styles=0,1
Arial.Type=1

DefaultFont.Family=Arial
DefaultFont.Style=0
DefaultFont.Size=8.0
```

Below is an example of the same file after a new font, Batang, was added. For visualization purposes the added font lines have been colored **red**.



Remember to verify that the font is installed in the Operating System.

```
#
#Wed Jun 23 11:29:33 EDT 2004
bitmappedFont=1
scalableFont=1

InstalledFontFamilies=SansSerif,Arial,Batang

SansSerif.Sizes=8.0,10.0,12.0,14.0,16.0,18.0,20.0,22.0,24.0
SansSerif.Styles=0,1,2,3
SansSerif.Type=2

Arial.Sizes=8.0,12.0,16.0
Arial.Styles=0,1
```

```
Arial.Type=1
Batang.Sizes=8.0,10.0,12.0,14.0,16.0,18.0,20.0,22.0,24.0
Batang.Styles=0,1,2,3
Batang.Type=2
DefaultFont.Family=Arial
DefaultFont.Style=0
DefaultFont.Size=8.0
```



Uninstalling a Font from the Emulator Printer

Use

To uninstall a font from the Emulator you must manually edit the file called *InstalledFont.username*.



Refer to the section called [Installing a font in the Emulator Printer \[Seite 56\]](#) for the overall file structure.

Procedure

1. Make a backup copy of the file named *InstalledFont.username*, where username refers to the logon name of the user.



The developer can change the default file by selecting the *InstalledFont.dft* instead of *InstalledFont.username*. Delete the file named *InstalledFont.username* for these changes to be reflected in the configuration of the developer.



Changes made to the default file affect any user who launches the Emulator for the first time after these changes are done.

2. Open the file named *InstalledFont.username* in a text editor.
3. Remove the font family name from the list of supported font family names and delete all the applicable font detail lines.



If the default font family is the one to be deleted, modify the default font to an available font family.



The Printer Emulator will not perform properly if all fonts are uninstalled.

Result

The font is uninstalled.



Installing New Media to the Emulator Printer

Use

To add media to the Emulator you must edit the file named *SupportedMedia.username*.

This file stores the following information:

- Media capabilities of the emulated printer.
- Installed media and its details.
- Current default media settings.

A description of the file is given below:

Key	Value Description
CurrentPrintHeadWidth	It is set to the value of the selected print head width. This value must be listed in SupportedPrintHeadWidth. This value can also be set via the Printer <i>Capability...</i> window.
Continuous	1 means the emulated printer will use continuous paper. 0 means the emulated printer will use non-continuous paper. The emulated printer can only have one type of media active at any given time. This value can also be set via the Printer <i>Capability...</i> window.
PaperLength	This value is used to set the length of the continuous media. It can also be set via the Printer <i>Setup...</i> window. The valid range for this key starts at 163 cm/64 inches and ends at 1626 cm/640 inches.
SupportedMedia	Lists the media currently supported by the emulator. Any media that is added to the emulator should be added here.
CurrentMedia	It is set to the name of the current media loaded. This value can also be set via the Printer <i>Setup...</i> window.
SupportedPrintHeadWidth	Lists all the print head widths the emulator supports.
PaperCount	The number of pages for non-continuous media. This key supports a minimum value of 1 and goes up to 1000. This value can also be set via the Printer <i>Setup...</i> window.
PaperLengthDispalyUnit	Sets the unit to be used for the paper length. This can have one of two values: 0 means that paper length will be given in inches. 1 means that paper length will be given in centimeters. This value can also be set via the Printer <i>Setup...</i> window.

Media details: (for a media named *medianame*)

Key	Value Description
<i>medianame.Size</i>	<p>Description: This value contains the media dimensions in inches.</p> <p>Required: Yes</p> <p>Format: The format of this key is "width, height". The unit of measure for this value is "inches".</p> <p>Further Notes: It is important to know that continuous media has zero as its length.</p>
<i>medianame.Margin</i>	<p>Description: Holds the margin between printable area and the edge of the media.</p> <p>Required: No (optional)</p> <p>Format: It is specified in format of "top, left, bottom, right" and the unit of measure is "inches".</p> <p>Further Notes: Although it is visible on the virtual printout area, it is not enforced by the emulated printer. It is just a visual help.</p> <p>This key is only supported for non-continuous media.</p>
<i>medianame.Preprint</i>	<p>Description: Used for predefined areas settings in the media, like labels in the media, for example.</p> <p>Required: No (optional)</p> <p>Format: It is specified in format of "columns, rows, horizontal gap, vertical gap" in the unit of measure of "inches". The columns and rows entries must be integer values.</p> <p>Further Notes: Like the margin, it is just another visual help. The Emulator will calculate the width of the column and the height of the row.</p> <p>This key is only supported for non-continuous media.</p>

Procedure



The developer can modify the default configuration by editing the file named *SupportedMedia.dft*. For the changes to take effect in the developer's emulator, delete the file named *SupportedMedia.username*. The next time the Emulator opens it will be configured based on the default file.



Changes made to the default file affect any user who launches the emulator for the first time after these changes are done.

Result

The media has been added to the emulator printer.

Example

The following is a sample of the file. The sequence of lines has been adjusted and blank lines are added for ease of reading.

```
#
#Thu Jul 15 11:33:43 VET 2004
continuous=0
CurrentMedia=A4
CurrentPrintHeadWidth=8.5
PaperCount=100
PaperLength=320
PaperLengthDispalyUnit=0
SupportedMedia=Letter,Legal,A3,A4,B4,B5,Plain11x17,Plain4x8,2x4_in_4x
8,Continuous_1,Continuous_2,Continuous_3,Continuous_4,Continuous_8.5
SupportedPrintHeadWidth=1,2,3,4,8.27,8.5,12

2x4_in_4x8.Margin=0.2,0.2,0.2,0.2
2x4_in_4x8.Preprint=2,4,0.05,0.05
2x4_in_4x8.Size=4,8

A3.Size=11.69,16.54
A4.Size=8.27,11.67
B4.Size=10.12,14.33
B5.Size=7.17,10.12
Continuous_1.Size=1,0
Continuous_2.Size=2,0
Continuous_3.Size=3,0
Continuous_4.Size=4,0
Continuous_8.5.Size=8.5,0
Legal.Size=8.5,14
Letter.Size=8.5,11
Plain11x17.Size=11,17
Plain4x8.Margin=0.2,0.2,0.2,0.2
Plain4x8.Size=4,8
```

Below is the file after the new media has been added. For visualization purposes the changes are colored **red**.

```
#
#Thu Jul 15 11:33:43 VET 2004
continuous=0
CurrentMedia=A4
```

```
CurrentPrintHeadWidth=8.5
PaperCount=100
PaperLength=320
PaperLengthDispalyUnit=0
SupportedMedia=Letter,Legal,A3,A4,B4,B5,Plain11x17,Plain4x8,2x4_in_4x
8,Continuous_1,Continuous_2,Continuous_3,Continuous_4,Continuous_8.5,
New Media
SupportedPrintHeadWidth=1,2,3,4,8.27,8.5,12

2x4_in_4x8.Margin=0.2,0.2,0.2,0.2
2x4_in_4x8.Preprint=2,4,0.05,0.05
2x4_in_4x8.Size=4,8

New\ Media.Margin=0.2,0.2,0.2,0.2
New\ Media.Preprint=2,4,0.05,0.05
New\ Media.Size=8,10

A3.Size=11.69,16.54
A4.Size=8.27,11.67
B4.Size=10.12,14.33
B5.Size=7.17,10.12
Continuous_1.Size=1,0
Continuous_2.Size=2,0
Continuous_3.Size=3,0
Continuous_4.Size=4,0
Continuous_8.5.Size=8.5,0
Legal.Size=8.5,14
Letter.Size=8.5,11
Plain11x17.Size=11,17
Plain4x8.Margin=0.2,0.2,0.2,0.2
Plain4x8.Size=4,8
```



Uninstalling Media from the Emulator Printer

Use

To uninstall media from the emulator the file *SupportedMedia.username* must be edited manually.



Refer to [Installing New Media to the Emulator Printer \[Seite 60\]](#) for an overview of the structure of the file.

Procedure

1. Make a backup copy of the file named *SupportedMedia.username*, where username refers to the logon name of the user.



The developer can change the default file by selecting the *SupportedMedia.dft* instead of *SupportedMedia.username*. Delete the file named *SupportedMedia.username* for these changes to be reflected in the configuration of the developer.



Changes made to the default file affect any user who launches the Emulator for the first time after these changes are done.

2. Open the file named *SupportedMedia.username* in a text editor.
3. Remove all detail lines for the media to be removed.
4. Remove the media name from the list of supported media.
5. If the current media name is no longer in the list, set it to one that is still available.
6. The Printer Emulator will not perform properly if all media is uninstalled.

Result

The media is uninstalled.



Installing a Font Configuration

Use

The Client API uses on a font configuration file containing a list of all fonts known by the API. These fonts must be supported by the physical/emulated printer. Installing a new font on the Emulator (Refer to [Installing a Font in the Emulator Printer \[Seite 56\]](#)) is analogous to installing a font on a physical printer. The font configuration file must be modified before the Client API recognizes the new font. Configuration changes to the drivers are done via the MI Configuration Tool. Configuration changes to the emulator are done by manual modification of the configuration files.

The file named *pipremulator.fntcfg* must be edited to modify the emulator font configurations. This file contains the name, description, options, font type, and size. Even though the file can be edited programmatically, a developer may decide to modify the file manually for the emulation of one specific printer hardware.

This file is located in:

```
ECLIPSE_HOME\plugins\com.sap.ip.me.mdk.pios.docgen_X.Y.Z\emulator\pios\  
config\pipremulator.fntcfg
```


Where ECLIPSE_HOME refers to the location in the hard disk where Eclipse is installed and X.Y.Z refers to the plugin version, for example 1.1.0.



This file does not exist the first time the emulator runs. It is created and read when the Printer API opens the connection to the Emulator.



This file is read during the opening of the connection to the printer. If a change is made after the connection has been opened and before the connection is closed, changes will take effect the next time the connection opens.

File structure:

File Variables

Variable	Description
Configs	Lists the font configurations available to the PIOS Client API.

Fonts Parameters

A font configuration called *fontcfgrname* needs the options explained below. Font configurations must be listed in the Configs variable mentioned above.

Parameter	Description
<i>fontcfgrname</i>	This is left blank, but the line must be included in the file.
<i>fontcfgrname._Type</i>	This parameter is always set to "Font".
<i>fontcfgrname.Name</i>	The name of the font.
<i>fontcfgrname.Description</i>	Parameter that contains a short description of the font.
<i>fontcfgrname.Options</i>	Specify the options of the font. Possible values are: 0 - Normal 2 - Bold 4 - Italic 6 - Bold and italic 8 - Underline 10 - Bold and underline 12 - Italic and underline 14 - Bold, italic, and underline
<i>fontcfgrname.FontType</i>	Tells whether the font type is scalable or bitmapped. Possible values for this parameter are: 1 - bitmapped 2 – scalable
<i>fontcfgrname.Size</i>	This value tells the size of the font.



Make sure font configurations added to this file match those installed in the *InstalledFont.username*. The font configurations included in this file map installed fonts in the emulator.

Procedure

1. Make a backup copy of the file named *pipremulator.fntcfg*.
2. Open the file named *pipremulator.fntcfg* in a text editor.
3. Add the font family name to list of supported configurations (*Configs*). (Refer to before and after example below.)
4. Add the font configuration parameter lines: *_Type*, *Name*, *Description*, *Options*, *FontType*, and *Size*. (Refer to before and after example below.)

Result

Font configuration is added.

Example

The following is an example of font configuration file. The sequence of lines has been adjusted and blank lines are added for ease of reading.

```
#Fri Jul 16 16:02:53 VET 2004
#Initial Installation

Configs=Bitmapped

Bitmapped=
Bitmapped._Type=Font
Bitmapped.Name=Courier New
Bitmapped.Description=plain bimapped font
Bitmapped.Options=0
Bitmapped.FontType=1
Bitmapped.Size=10
```

Below is an example of the same file after a new font configuration, *ScalableBoldItalic*, is added. For visualization purposes the added font lines have been colored **red**.

```
#Fri Jul 16 16:02:53 VET 2004
#Initial Installation
Configs=Bitmapped,ScalableBoldItalic
Bitmapped=
Bitmapped._Type=Font
Bitmapped.Name=Courier New
Bitmapped.Description=plain bimapped font
```

```
Bitmapped.Options=0
Bitmapped.FontType=1
Bitmapped.Size=10

ScalableBoldItalic=
ScalableBoldItalic._Type=Font
ScalableBoldItalic.Name=Batang
ScalableBoldItalic.Description=bold&italic scalable font
ScalableBoldItalic.FontType=2
ScalableBoldItalic.Options=6
ScalableBoldItalic.Size=10
```



Uninstalling a Font Configuration

Use

A font configuration can be uninstalled programmatically but a developer may decide to do it manually. To uninstall a font configuration manually from the Emulator you must manually edit the file called *pipremulator.fntcfg*.



Refer to the section called [Installing a Font Configuration \[Seite 64\]](#) for the overall file structure.

Procedure

1. Make a backup copy of the file named *pipremulator.fntcfg*.
2. Open the file named *pipremulator.fntcfg* in a text editor.
3. Remove the font family name from the list of supported font family names and delete all the applicable font configuration parameter lines.



The Printer Emulator will not perform properly if all font configurations are uninstalled.

Result

The font configuration is uninstalled.



Using the PIOS Emulator for the Printer Peripheral







Use


Provide an overview on how to use the Peripheral I/O Emulator for the printer peripheral type.

Prerequisites

- SAP NetWeaver Developer Studio is installed and configured.

Procedure

1. Write code that uses the Printer API of the MI Client API.
2. Once the code is ready for testing, set the SAP NetWeaver Developer Studio in emulation mode by pressing the *Set MDK peripheral emulation mode*  button on the MI perspective.
3. Restart the Mobile Infrastructure client.
4. Launch the Emulator by pressing the *Launch the Peripheral I/O Emulator*  button on the MI perspective.
5. Turn the Emulator offline by clicking on the *On/Offline*  button.
6. Configure desired emulator capabilities by clicking on the *Capability...*  button.
 - a. Under the general tab, select the desired capabilities to be emulated.
 - b. Under the barcode tab, select the desired symbology capabilities to be emulated.
 - c. Under the font tab, select the desired font capabilities to be emulated.
 - d. Under the media tab, select the desired media *Type* and *Print Head Width* to be emulated.
7. Configure the appropriate Emulator settings by clicking the *Setup...*  button.
 - a. Select the media that the printer will emulate in the Virtual Printout Area and set the Paper Count/Page Length.
 - b. Set the default font. This is only used for the barcode's Human Readable attribute.
 - c. Set the printing resolution for the emulator printer.
8. Set the emulator Online by clicking on the *On/Offline*  button.



Make sure the Emulator is Online by checking the *Status* property on the *State Buffer*.
9. Run your application.

Result

The application print output is displayed on the virtual printout area.



Using the Measuring String

Use

The Measuring String is used to determine the distance between two points inside the Virtual Printout Area. This distance is measured in points. It displays two numbers, the horizontal and vertical spacing.

Prerequisites

- The printer peripheral panel is displayed.

Procedure

1. Move the mouse over the Virtual Printout Area, press and hold the left mouse button over the first point, drag the mouse to the next point. A line appears between the first point and actual mouse pointer position.
2. While the mouse button remains depressed, the horizontal and vertical distance between the two points is displayed.

Result

The Measuring String displays the horizontal and vertical distance between the two points selected.



Driver Selection Tool

Purpose

The Driver Selection Tool (DST) enables the SAP MI Web Console administrator select peripheral add-on(s) that meet the mobile application peripheral requirements. This selection process also considers the mobile application target device operating system, virtual machine, processor and available transports.

Integration

The DST is integrated into the SAP MI Web Console.

Features

- **Display Matched Drivers**

Displays available drivers that match the target OS, processor, VM, transport and application requirements.

- **Display Non-Matched Drivers**

Displays available drivers that do not match the target OS, processor, VM, transport and application requirements. It also displays the first selection criteria that is not met.

Constraints

- The DST only recommends driver add-ons registered in the DST driver catalog. The catalog is updated via a Service Pack installation.
- The DST only displays the first reason for a mismatch. It may display the operating system, transports, or attributes in that order. If the reason for a mismatch is the attributes, it will show all the required attributes that were not matched.



Using the Driver Selection Tool


Use

The Driver Selection Tool is intended for the system administrator. It describes the process of launching the Driver Selection Tool and searching for drivers that match selected requirements.

Prerequisites

- You started the SAP MI Web Console (see [Starting the SAP MI Web Console \[Externl\]](#)).
- Mobile application with a Driver Requirements Document is available in the SAP Web Console

Procedure

1. Identify an application in the SAP Web Console that requires peripheral support.
2. Choose *Select driver*  on the right side of the mobile application identified.
The Driver Selection Tool (DST) is launched.
3. Select the target operating system, processor, virtual machine, and transports.

Transport - In this field you select one or more transports. The transport is part of the target mobile device. It enables you to establish the connection between the mobile device and the peripheral hardware.



When more than one transport is selected, the DST searches for at least one of them. In other words, it follows a logical OR behavior. When at least one of the transports is found to be supported by the driver the requirement is considered met.

4. Click on *Search*. The list of matching drivers or the *No Matching Drivers* warning window is displayed.
5. If the *No Matching Drivers* warning window appears, go back to step 3 and change the selection.

Result

A list of matching drivers is displayed.