

# Using the Printable Document

## 10.2

March 2024

### DOCUMENT ACCESS

Public

### DISCLAIMER

The contents of this document are under copyright of Critical Manufacturing S.A. it is released on condition that it shall not be copied in whole, in part or otherwise reproduced (whether by photographic, or any other method) and the contents therefore shall not be divulged to any person other than that of the addressee (save to other authorized offices of his organization having need to know such contents, for the purpose for which disclosure is made) without prior written consent of submitting company.

# Using the Printable Document

*Estimated time to read: 9 minutes*

During the manufacturing process it may be necessary to print labels and documents. This can be achieved by using **Printable Document**.

The **Printable Document** entity is the core object of Advanced Layout and Printing, which is the module that provides the ability to design and print labels, lot travelers and other types of documents, such as Material Identification or a Certificate of Conformance.

## Info

Advanced Layout and Printing is a Critical Manufacturing optional module.

This document provides step-by-step information so you can accomplish some common **Printable Document** scenarios.

## Overview

The **Printable Document** represents a generic document whose content is generated dynamically whenever it is printed, based on a context that is a combination of user inputs and automatically calculated data (including serial numbers).

It also ensures full traceability by versioning the **Printable Document** object and, optionally, keeping the history of each print operation including a preview of what was printed.

The **Printable Document** is available for every object.

## Creating a Material Document

This section describes how to create a simple document for a **Material**. The example will illustrate the usage of expressions and DEE Actions for calculating dynamic content.

## Info

The procedure below is similar for every object type. The only difference is that for **Material**, printing can be initiated from the Material Details page, the Step View and the Resource View; for all other object types, the actual printing needs to be initiated from the **Printable Document** details page.

## Configure the Printer Types

1. Add the different Printer Types to the Lookup table called *PrinterType*.
2. Configure the Smart Table *PrinterTypePrinterContexts*. This table maps the different Printer Types to specific Printers for particular users and computers.

Printer Type Printers Context

Refresh Add More

Printer Type Printers Context (Active)

PRINTER TYPE: ALL User Name: Computer Name: Advanced

Printer Type Printers Context (2)

	PRINTER TYPE	USER NAME	COMPUTER NAME	STEP	RESOURCE	RESOURCE TYPE	MODEL	PRINTER NAME
<input type="checkbox"/>	General			Packing	Packer-02			OfficePrinter
<input type="checkbox"/>	General			Packing	Packer-01			HeatTransfer

## Configure the Necessary DEE Actions

### Info

This is an optional step. The recommendation is to use DEE Actions only if it's not possible to achieve the same result using expressions or the expression editor of the layout designer.

In this example, a simple action will be used to read the attribute *InspectionRate* of the Material and concatenate it with the *CustomerName* attribute of the **Material Product**. The Customer Name is associated with the non-versioned portion of the **Product**.

## Create the DEE Action

Here is a DEE Action sample, called *PrintableDocumentGetInspectionRateAndCustomer*:

```
UseReference("Cmf.Foundation.BusinessObjects.dll", "Cmf.Foundation.BusinessObjects");
UseReference("Cmf.Navigo.BusinessObjects.dll", "Cmf.Navigo.BusinessObjects");
UseReference("Cmf.Foundation.BusinessOrchestration.dll", "");
UseReference("", "Cmf.Foundation.Common.Exceptions");
UseReference("", "Cmf.Foundation.Common");

IMaterial mat = Input["AppliesToValue"] as IMaterial;

IProduct prod = mat.Product;

//Load Attributes
mat.LoadAttributes(new Collection<String>() { "InspectionRate" });

prod.LoadAttributes();

Dictionary<String, Object> Output = new Dictionary<string, object>();

Output["Result"] = String.Format("{0}% - {1}",

mat.Attributes.ContainsKey("InspectionRate") ? mat.Attributes["InspectionRate"] : "",

prod.RelatedAttributes.ContainsKey("CustomerName") ? prod.RelatedAttributes["CustomerName"] : "");

return Output;
```

To make the DEE Action available for **Printable Documents**, it's necessary to:

1. Define the Scope Classification for DEE actions by editing the Generic Table *ScopeClassifications* and creating an entry *PrintableDocumentContextItem*

Scope Classifications

Refresh Add More

General Table Data Actions

Scope Classifications (Active)

SCOPE: ALL Advanced

Scope Classifications (10)

SCOPE	CLASSIFICATION
<input type="checkbox"/> EFC	EFC
<input type="checkbox"/> AlarmManagement	AlarmManagement
<input type="checkbox"/> CertificationManagement	CertificationManagement
<input type="checkbox"/> LaborManagement	LaborManagement
<input type="checkbox"/> SPCAction	SPCAction
<input type="checkbox"/> ExceptionManagementAction	ExceptionManagementAction
<input type="checkbox"/> ConnectToT	ConnectToT
<input type="checkbox"/> PrintableDocumentContextItem	PD

Rows per page: 10 Page 1 of 1 (10 records)

2. Create the DEE Action with the code specified above.

PrintableDocumentGetInsp...

Refresh Save Set Effective Disable Validate Import Delete More

Entity DEE Action Actions

PrintableDocumentGetInspectionRateAndCustomer.1 (Active)

Action code

```

1 UseReference("Cmf.Foundation.BusinessObjects.dll", "Cmf.Foundation.BusinessObjects");
2 UseReference("Cmf.Navigo.BusinessObjects.dll", "Cmf.Navigo.BusinessObjects");
3 UseReference("Cmf.Foundation.BusinessOrchestration.dll", "");
4 UseReference("", "Cmf.Foundation.Common.Exceptions");
5 UseReference("", "Cmf.Foundation.Common");
6
7 Material mat = Input["AppliesToValue"] as Material;
8 Product prod = mat.Product;
9
10 //Load Attributes
11 mat.LoadAttributes(new Collection<String>() { "InspectionRate" });
12 prod.LoadAttributes();
13
14 Dictionary<String, Object> Output = new Dictionary<string, object>();
15
16 Output["Result"] = String.Format("{0}% - {1}",
17     mat.Attributes.ContainsKey("InspectionRate") ? mat.Attributes["InspectionRate"] : "",
18     prod.RelatedAttributes.ContainsKey("CustomerName") ? prod.RelatedAttributes["CustomerName"] : "");
19
20 return Output;
21

```

TEST CONDITION CODE

INPUT PARAMETERS

3. When creating the DEE action (like the one with the source code example above), set its classification to the value defined in the previous point.

PrintableDocumentGetInsp

Refresh
Disable
Execute
Delete
More

Entity
DEE Action
Actions

{} PrintableDocumentGetInspectionRateAndCustomer.3 (Effective)

### General Data

{}

Name: PrintableDocumentGetInspectionRateAndCustomer  
Description:  
Classification: PD  
Enabled: ☒ Yes  
Universal State: Effective

4. Create a **Rule** object with scope *PrintableDocumentContextItem* and referring to the entry created in point 1.

\* Create Rule

GENERAL DATA

Name: PrintableDocument\_GetPD  
Description:  
Data Group:

Information

\* Scope: PrintableDocumentContextItem  
\* DEE Action: {} PrintableDocumentGetInspectionRateAndCustomer

The possible DEE Rules must match the classifications defined in the generic table ScopeClassification for the current Rule scope

Comments:

Cancel Create

## Create the Printable Document

1. Assuming that there is a **Change Set** created, create the **Printable Document**:

Create Printable Document

1 CHANGE SET

2 GENERAL DATA

3 DATA CONTEXTS

General Data

Name:

SampleMaterialDocument

Description:

Type:

Label

Data Group:

Scope:

Label

Applies to:

Material

Properties

Store Print History:

Store Layout History:

Print Settings

Printer Type:

General

Default Printer:

OfficePrinter

Comments:

Cancel

Back

Next

2. You can also edit the **Printable Document**:

Edit Printable Document

GENERAL DATA

DATA CONTEXTS

General Data

Name:

SampleMaterialDocument

Description:

SampleMaterialDocument

Type:

Label

Change Set:

AA\_08

Data Group:

Scope:

Label

Applies to:

Material

Properties

Store Print History:

Store Layout History:

Print Settings

Printer Type:

General

Default Printer:

Comments:

Cancel

Save

#### Note

If your **Printable Document** is in the Created state, you will be able to edit more properties than when it is in the Effective state.

**Edit Printable Document**

GENERAL DATA DATA CONTEXTS

Data Contexts

- Duration  
No Description
- Quantity  
No Description
- Step  
No Description

Data Context Details

\* Name: Duration

Description:

\* Type: Input

Is Serial Number: ☐

\* Value Data Type: Decimal

Value Object Type:

Value Is Collection: ☐

Default Value:

\* Source Type: Free

Source:

Comments:

Cancel Save

### Info

For more information, see [Create Printable Document](#) and [Edit Printable Document](#) in the User Guide.

- Next, in the Layout designer, edit a page layout. You may reference any property of the `$AppliesToValue` entries or use the variables (PrimaryQuantity, FacilityName or TestLabel) that were used in this case. Be sure to save the layout before closing the layout editor.

SampleMaterialDocument.1 (Editing Layout)

Home Insert Page Layout

Clipboard Font Alignment Borders Text Format Conditions Copy Style Style Designer Select Style

Dictionary

Actions

- Data Sources
- Variables
- PrimaryQuantity
- FacilityName
- TestLabel
- NumberOfCopies
- PrinterName
- UserName
- ViewMode
- System Variables
- Functions
- Resources

Properties Dictionary

Page1

Quantity: {PrimaryQuantity}

Location: {FacilityName}

Inches Check for Issues DataBand2 X:0.00 Y:0.20 Width:7.70 Height:10.20 70%

### Info

For more information on how to create and edit the page layout, see [Printable Document Layout Creation](#) in the User Guide.

## Print the Document

After setting the document effective, the document can be printed directly from the **Printable Document** details page.

Print Printable Document

1

CONTEXT

2

PRINTER

SampleMaterialDocument.1

Applies To

\*Material:

Planned Material

x

Data Contexts

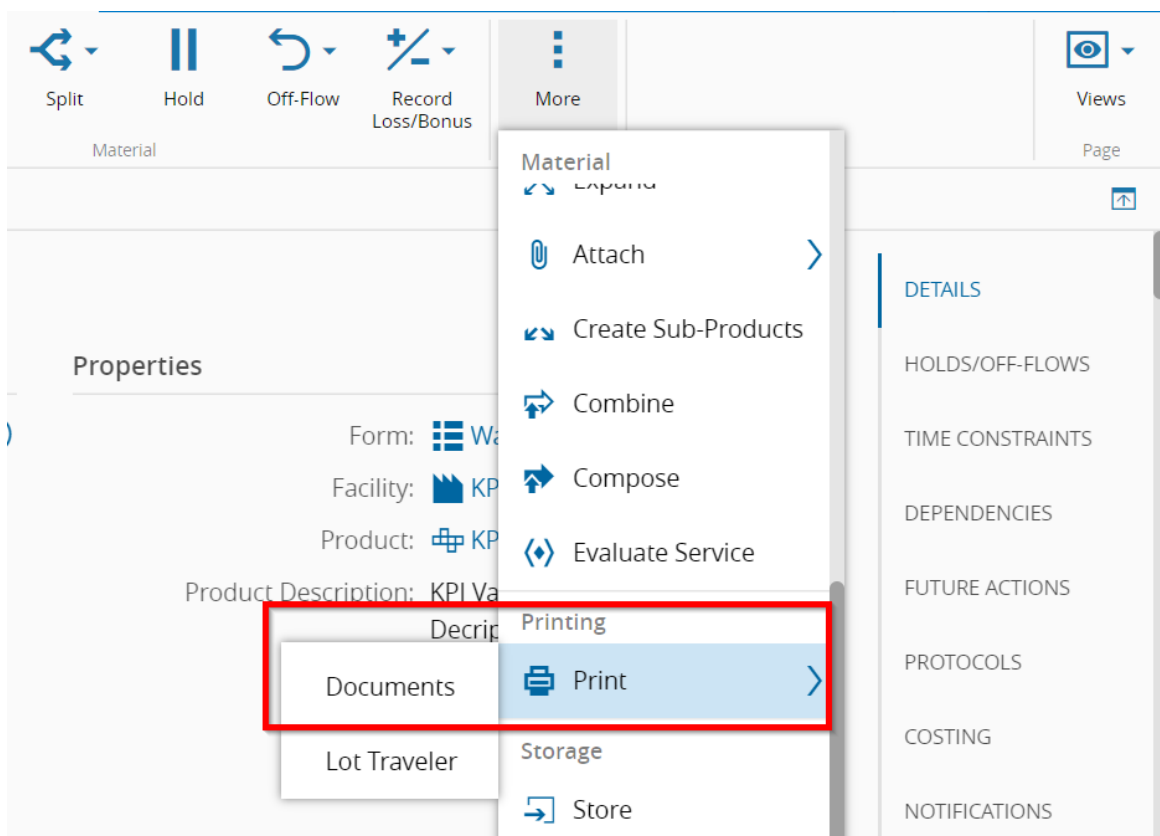
Comments:

Cancel

< Back

Next >

In order to be able to print the document directly from the **Material** (in alternative, from the Step View or Resource View), it's necessary to define the context in the Smart Table [MaterialPrintableDocumentContext](#). An example is shown in the next picture:



When selecting the **Print Documents** option, all **Documents** matching the document context will be available for the user.

## Printing a Lot Traveler

The Lot Traveler is constructed dynamically based on the next **Steps** and **Flows** of the **Material**.



### Warning

The Lot Traveler will print the **Material** current **Step** and all the next non-Optional, non-Alternate **Steps** for the **Area**.

### Warning

If a **Step** of the **Area** does not have a Lot Traveler defined, the system will try to use the default Lot Traveler document from the Smart Table [AreaPrintableDocumentContext](#). If no default document exists, the **Step** will be skipped in the Lot Traveler printing.

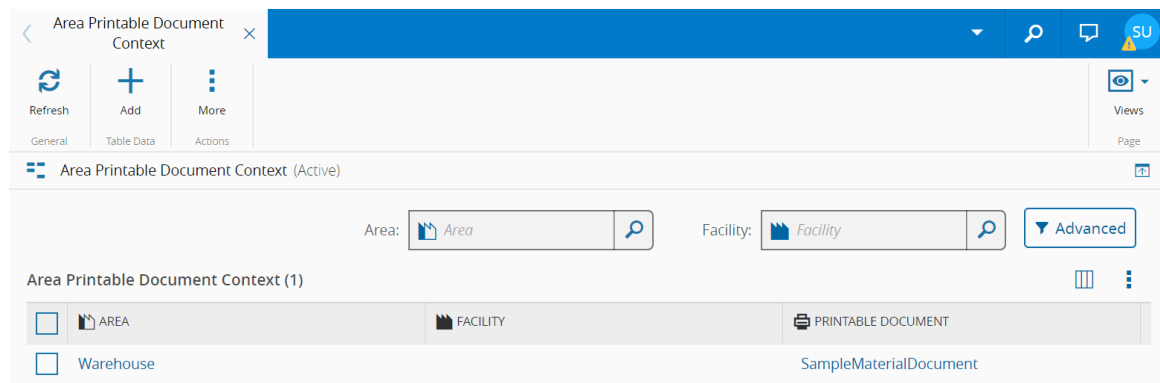
## Create the Lot Traveler

The creation process of a **Printable Document** suitable for being used in a Lot Traveler is similar to the creation of **Printable Document** as explained above with the particularity that, it must be of type "Lot Traveler". When a **Printable Document** is defined of type "Lot Traveler", the Applies To property is automatically set to Material. In addition, three Data Contexts are automatically created: **Step**, **Flow** and **Flow Path**. Because the Lot Traveler is based on dynamic information of the **Material**, during runtime, the system will supply the appropriate values for these three contexts for each **Step**.

## Associate the Lot Traveler with the Steps

Each **Step** can have its own Lot Traveler document, or each **Step** can have its own unique Lot Traveler document. To associate a Lot Traveler with a **Step**, it's necessary to edit the **Step** and set the **Step Lot Traveler** property.

For convenience, it's also possible to define the Lot Travelers for a complete **Area** or **Facility** in the Smart Table [AreaPrintableDocumentContext](#). An example is shown in the next picture:



The screenshot displays the 'Area Printable Document Context' Smart Table. The table has three columns: 'AREA', 'FACILITY', and 'PRINTABLE DOCUMENT'. The first row shows 'Warehouse' in the 'AREA' column and 'SampleMaterialDocument' in the 'PRINTABLE DOCUMENT' column. The second row is partially visible, showing 'AREA' and 'FACILITY' columns.

To print the Lot Traveler (it always prints the Lot Traveler for the complete **Area**), it's just necessary to select the desired **Material** in the Step View, Resource View or in the **Material** details page, and then select **Print Lot Traveler**.

Print Lot Traveler

FQ\_Material\_3 (Queued) / FQ\_Product / FQ\_Step\_1 / 1000 Kg

PRINTER

PREVIEW

Print Options

\* Printer: GenericPrinter

\* Lot Traveler Scope: CurrentArea

Cancel

Print

#### Info

By default, it only prints the Lot Traveler for all the current and next steps of the current **Area**. It's also possible to print the Lot Traveler for the current **Step** by selecting the *Current Step only* option on the right side of the Wizard or print the Lot Traveler for the all **Facilities** by selecting the All Facilities only option on the right side of the Wizard.

#### Info

More information on how to print a Lot Traveler or other **Material** documents can be found at [Print Material Documents](#) section of the User Guide.

## Additional information

### Using Entity Types in a Printable Document

**Printable Documents** can be created for any entity type by defining the desired entity type as *Applies To* property. A print button on the entity details page appears only for **Material** objects and documents to be printed for a given **Material** are resolved using the Smart Table [MaterialPrintableDocumentContext](#). For all other EntityTypes, printing has to be triggered from the **Printable Document** page.

### Linking a dynamic variable to a Entity Type property

There are two possible options:

1. Define a ContextItem for which the value is an EntityInstance (Object) and in the Document designer, link the layout field values to the desired property of that object - when the value of a ContextItem is an **EntityType**, the designer will show all properties of that **EntityType**, so any property can be used. This is also applicable to the *AppliesTo* property, but in this case it appears as a parameter named `AppliesToValue` on the designer.
2. Define a ContextItem that uses an expression to extract the value of the desired property. In this case the designer will be provided with property value directly instead of the entire Entity Instance with all its different properties. The expression for this would be in the format: `$(ParameterName).(PropertyName)` where `ParameterName` is the name of the parameter holding the

EntityInstance (may be another ContextItem or the `$AppliesToValue` special parameter) and (PropertyName) is the name of the desired property.

#### Info

If the *AppliesTo* refers to a **Material** object, you can dynamically get the **Facility** Name by creating a Context Item of type expression and specifying the source as `$AppliesToValue.Facility.Name`.

## Linking a variable to an entity type attribute

In the case of attributes it becomes necessary to use a ContextItem that is calculated using a DEE Action since the Attributes must be loaded beforehand in order to be used. This would be a very simple DEE Action that just executes `LoadAttributes()` on the desired entity instance (possibly coming from another ContextItem or `$AppliesToValue`, as the DEE Action receives all values already calculated as input) and then returns the value of the desired attribute.

## String array support

In order to support `string[]` types, the ContextItem would be defined as being of Type `String` and the *Value is Collection* flag must be set to True. Then the DEE Action that extracts the value of the attribute needs to convert the attribute value (which will be `List<String>`) to a `Collection<String>`.

Example:

```
return new Collection<String>(attrvalue as List<String>);
```



# Legal Information

## **Disclaimer**

The information contained in this document represents the current view of Critical Manufacturing on the issues discussed as of the date of publication. Because Critical Manufacturing must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Critical Manufacturing, and Critical Manufacturing cannot guarantee the accuracy of any information presented after the date of publication. This document is for informational purposes only.

Critical Manufacturing makes no warranties, express, implied or statutory, as to the information herein contained.

## **Confidentiality Notice**

All materials and information included herein are being provided by Critical Manufacturing to its Customer solely for Customer internal use for its business purposes. Critical Manufacturing retains all rights, titles, interests in and copyrights to the materials and information herein. The materials and information contained herein constitute confidential information of Critical Manufacturing and the Customer must not disclose or transfer by any means any of these materials or information, whether total or partial, to any third party without the prior explicit consent by Critical Manufacturing.

## **Copyright Information**

All title and copyrights in and to the Software (including but not limited to any source code, binaries, designs, specifications, models, documents, layouts, images, photographs, animations, video, audio, music, text incorporated into the Software), the accompanying printed materials, and any copies of the Software, and any trademarks or service marks of Critical Manufacturing are owned by Critical Manufacturing unless explicitly stated otherwise. All title and intellectual property rights in and to the content that may be accessed through use of the Software is the property of the respective content owner and is protected by applicable copyright or other intellectual property laws and treaties.

## **Trademark Information**

Critical Manufacturing is a registered trademark of Critical Manufacturing.

All other trademarks are property of their respective owners.