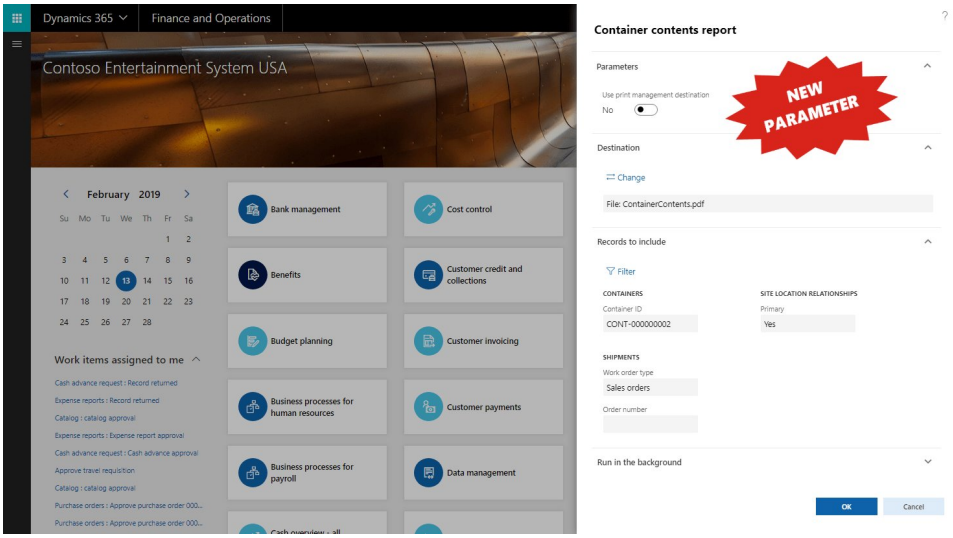
Add a new parameter to SSRS reports in Dynamics 365 for Finance and Operations

Posted on February 16, 2019 by Ana Gligorijevic

Tags: D365FO, Report parameters, SSRS 3 comments



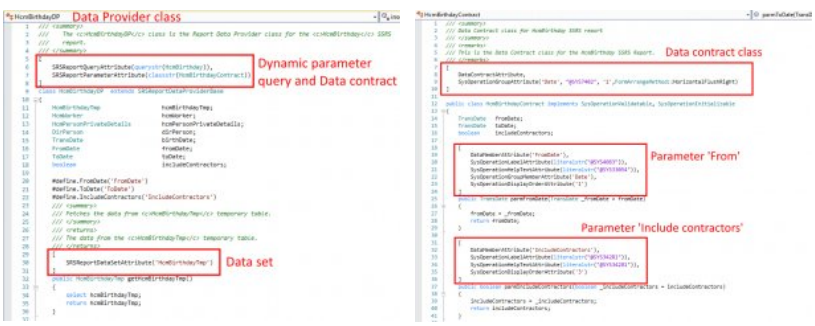
Before we explain how to add a new parameter to an SSRS report, let just remind ourselves first that there are two types of report data sets and therefore two types of SSRS reports in Dynamics 365 for Finance and Operations:

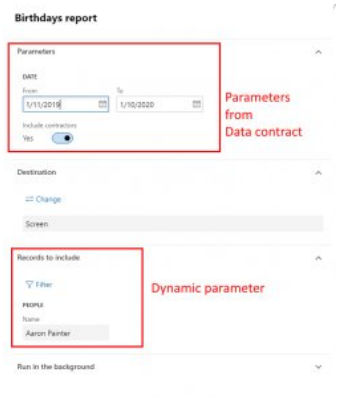
- Query based reports
- DP (Data Provider) based reports

The procedure of adding new parameters depends on this report type.

How to add a new parameter to a DP based SSRS report

A DP based SSRS report points to a DP class as its data source. A DP class fills temporary tables (actually InMemory, TempDB or Regular) using some business logic and exposes them (using `SRSReportDataSetAttributes`) to the related SSRS report as data sets. Also, a DP class points to a Data contract class, which defines and stores values of this SSRS report's parameters. And finally, a DP class can point to an AOT query which acts as Dynamic parameter. For example:





DP based SSRS report > DP class > Data contract (parameters), Dynamic parameter query, Data sets.

If we want to add a new parameter to a DP based report, we need to do this by adding it to the Data contract class. This was a simple procedure in AX 2012 but in Dynamics 365 for Finance and Operations, we must not use overlayering but only extensions. Unfortunately, we cannot simply extend Data contract class since extensions don't support the attributes making the link to the data contract and dynamic parameter query. So, we have to do it the hard way.

First, we need to create a new contract (to derive from the existing one) with a new parameter, and consequently to create a new DP (to derive from the existing one) that is related to the new contract through the `SRSReportParameterAttribute` attribute, and then to override the `processReport()` method in order to use that new parameter. Consequently, we need to duplicate the existing SSRS report and select the new DP class as its data source, and afterwards to extend the existing report menu item to point to the duplicated report. Sometimes, we will need to extend the controller class too and/or implement a Print management event handler; this depends on the report to which we are adding a new parameter.

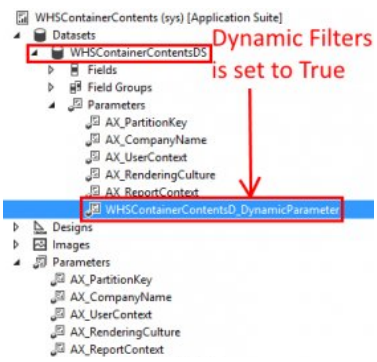
In order to see the newly added parameter under the **Parameters** node on the report in AOT, we need to assign the new DP class as a report data source. This will refresh the report's **Parameters**. Afterwards, in order to make the newly added parameter appear on the report dialog form, we need to compile and deploy the report.

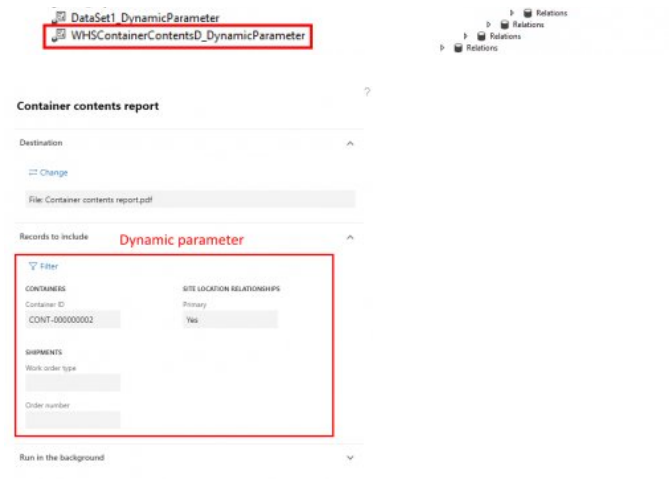
On how to duplicate a report, extend its menu item and controller class you can learn in more detail in the next chapter.

How to add a new parameter to a Query based SSRS report

A Query based SSRS report points to a query from AOT. A query defines one or more data sets while query ranges becomes report parameters within or outside Dynamic parameter.

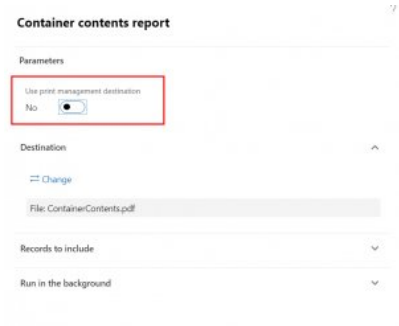
Query based SSRS report > Query > Query ranges (parameters), Dynamic parameter, Data sets.





> In the Why is Container contents report always printed to Screen in D365FO article we saw that **Container contents report** has a bug that manifests as ignoring the print destination selected in the report dialog and always using the print destination selected in **Print management setup** instead. This bug can be fixed by adding an additional report parameter **Use print management destination** as described below.

So, let's say that we want to add an additional report parameter **Use print management destination** outside Dynamic query which will not be a query range, but we will use it in the report controller class to change the current execution flow. Namely, depending on this parameter's value, either the print destination settings selected on the report dialog form or those from **Print management setup** will be used when the report is executed. After this parameter is added, it will appear on the report dialog form within the **Parameters** tab and not within the **Records to include** tab:

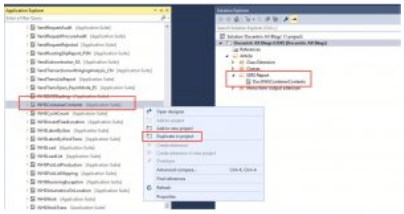


Navigation path to this report is: **Warehouse management > Inquiries & reports > Container contents report**.

Step 1: Duplicating the report

Since **Container contents report** is a Query based SSRS report it doesn't have a Data contract class to add a parameter. And since we don't want to use a query range to define the **Use print management destination** parameter, we will add it as a separate parameter directly to the **Parameters** node on the report as shown on the screenshot in Step 2.

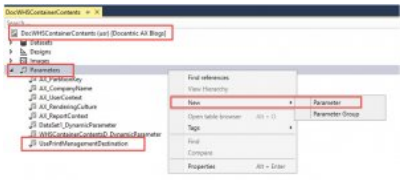
However, because we cannot use overlayering, we need to duplicate this report first and then to add the new parameter to it. We can duplicate a report by selecting it in **Application Explorer** and adding it to our Visual Studio project as shown below.

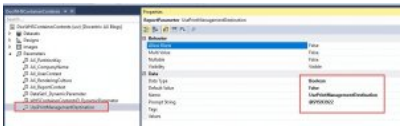


Step 2: Adding the parameter

We will now open our duplicated SSRS report in **Solution Explorer**. Expand and right-click on the **Parameters** node, choose **New** on the pop-up menu and then select **Parameter**. This will add a new parameter.

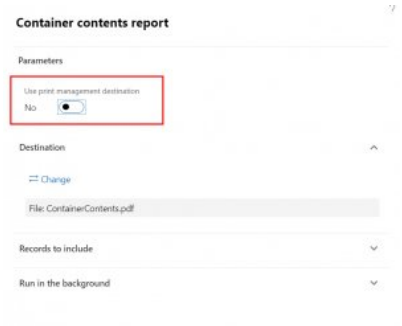
Use **Properties** to name the parameter as **UsePrintManagementDestination** and set its **Data Type** to **Boolean**, its **Default Value** to **False** and its **Prompt String** to **@SYS93922** ('Use print management destination')





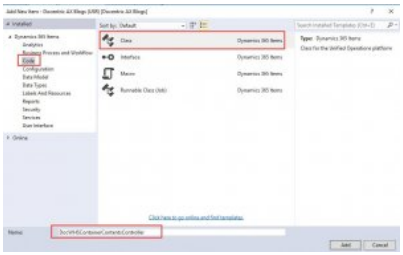
> ⚠ In order to make the new parameter appear on the report dialog form, we need to compile and deploy the report first.

After we complete Step 3 and Step 4, and run the report, the newly added parameter will appear on the report dialog form.



Step 3: Extending the controller class

Now we will create a new controller class `DocWHSContainerContentsController` that will extend `WHSContainerContentsController` class and implement its `main()` method in order to be able to use our custom report design. Also, in the same class we will override the `runPrintMgmt()` method to change the built-in behavior to always use those print settings configured in **Print management setup**.



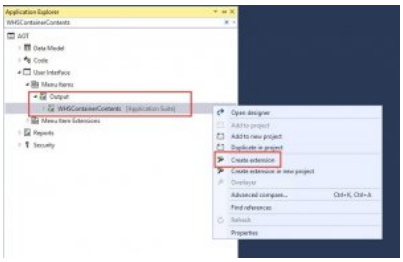
Add the following code to the new class:

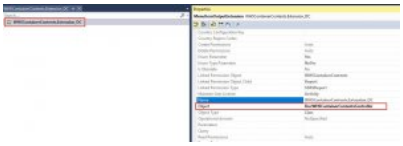
```
Class DocWHSContainerContentsController extends WHSContainerContentsController
{
    /// <summary>
    /// Main method to override report name
    /// </summary>
    /// <param name = "_args">Method arguments</param>
    public static void main(args _args)
    {
        //Declaration & Initialization of extended class
        DocWHSContainerContentsController controller = new DocWHSContainerContentsController();
```

```
        //Setting the custom report
        controller.parmReportName(ssrsReportStr(DocWHSContainerContents, Report));
        controller.parmArgs(_args);
        controller.startOperation();
    }


    /// <summary>
    /// Override this method to change the behavior to always use those
    /// print destination settings configured in Print management setup.
    /// </summary>
    protected void runPrintMgmt()
    {
        #define.UsePrintManagementDestination("UsePrintManagementDestination")

        // Get the UsePrintManagementDestination parameter value.
        boolean usePrintManagement = this.parmReportContract().parmRdlContract()
                                         .getParameter(#UsePrintManagementDestination)
                                         .getValueTyped();

        // If the parameter value is false, then we need to get and use the print
        // settings from the report dialog form and not from Print Management Setup.
        if (!usePrintManagement)
        {
            SRSPrintDestinationSettings pds = this.parmReportContract().parmPrintSettings();
            PrintMgmtReportRun printMgmtReportRunInstance = printMgmtReportRun;
            printMgmtReportRunInstance.parmDefaultOriginalPrintJobSettings(pds);

            // Force the use of the print settings selected on the report dialog.
            printMgmtReportRunInstance.parmForcePrintJobSettings(true);
        }

        // Call super().
        super();
    }
}
```

Step 4: Extending the report output menu item

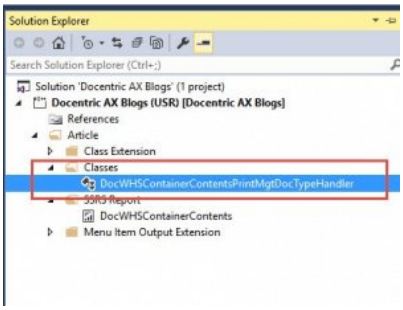
Now we will create an extension for the report output menu item (**WHSContainerContents**) in order to enable use of our custom controller class.





Step 5: Adding support for Print management setup

Since **ContainerContentsReport** report is a Print management report, we have to implement a delegate handler method that will enable our custom report design to be used with **Print management setup**.

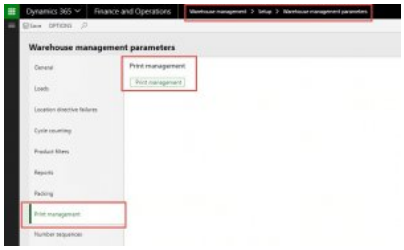


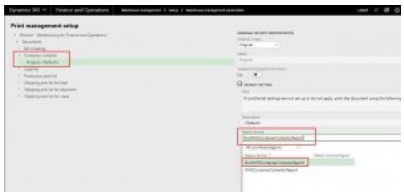
We need to subscribe to the `getDefaultReportFormatDelegate()` delegate placed in the `PrintMgtDocTypeHandlerExt` class and implement its handler method in the following way.

```
class DocWHSContainerContentsPrintMgtDocTypeHandler
{
    /// <summary>
    /// Add a subscriber method to set our custom SSRS design as a default SSRS report format
    /// for the WHSContainerContents PrintMgmtDocType but also to add it to Print management setup.
```

```
    /// </summary>
    /// <param name = "_docType">Print management document type</param>
    /// <param name = "_result">Event handler result</param>
    [SubscribesTo(classstr(PrintMgmtDocType), delegatestr(PrintMgmtDocType, getDefaultReportFormatD
    public static void getDefaultReportFormatDelegate(PrintMgmtDocumentType _docType, EventHandlerR
    {
        // For WHSContainerContents set our custom report design as the default report design.
        switch (_docType)
        {
            case PrintMgmtDocumentType::WHSContainerContents:
                // Our custom SSRS Design
                _result.result(ssrsReportStr(DocWHSContainerContents, Report));
                break;
        }
    }
}
```

After adding the delegate handler build your code and go to **Print management setup** (**Warehouse management > Setup > Warehouse management parameters > Print management** tab). You will be able to select our custom report design in the **Report format** combobox.





Result: Using the newly added report parameter

As you can see above, if we turn off the new parameter **Use print management destination** and select **File** as the target print destination on the report dialog form, the report will be printed to **File** and not to the print destination configured in **Print management setup**.

« Why is Container contents report always printed to Screen in D365FO

Add dynamic watermark to your reports in D365FO »

3 thoughts on “Add a new parameter to SSRS reports in Dynamics 365 for Finance and Operations”

Ayush Thapliyal says:

April 15, 2020 at 3:45 pm

Hi, Thanks for this blog. Explains a lot.

1 query that I have is can we use both AOT query and dynamic query in a Ssrs report so that both types of parameters appear in the parameter box.

If so, how? I’m unable to achieve this on my own.

Thanks in advance.

REPLY

Ana says:

April 15, 2020 at 6:04 pm

Dynamic query parameter that contains query ranges from the AOT query, which acts as the report dataset, will appear in the **Records to include** fast tab on the report dialog form, but only if you set the **Dynamic filters** property of the corresponding dataset to **True** before picking up the source query (for setting up the **Query** property of the dataset).

On the other hand, if you set **Dynamic filters** to **False**, and then select (once again) your AOT query as **Query** of the dataset, the query ranges will be added directly to **Parameters** of the SSRS report. This means that they will also show up in the **Parameters** fast tab on the report dialog form when the report is printed. I didn't test this myself but this is how it should work 😊

Kind regards,
Ana

REPLY

Ali Akdeniz says:

August 9, 2021 at 12:32 pm

Design – NewDesign, Copy all objects in OldDesign and paste in NewDesign, Delete OldDesign and Publish. This is my solution too, it works fine.

REPLY

Leave a Reply

Your email address will not be published. Required fields are marked *

Visual Text

Paragraph

Name *

Email *

(Address never made public)

Notify me of followup comments via e-mail. You can also click to subscribe without commenting.

POST COMMENT

Docentric respects your privacy. Learn how your comment data is processed >>

Subscribe to Our Blog

To get periodic updates on the
latest posts and new releases

Your name (Optional)

Your email (Required)

SUBSCRIBE

You can unsubscribe at any time. For information about how we handle your personal data, please read our privacy policy.

CATEGORIES

- Microsoft Dynamics AX (12)
  - SSRS Reports (11)
  - Word Templates (1)
- MS Dynamics 365 for Finance and Operations (144)
  - Docentric AX Free Edition (72)
  - Docentric AX Full Edition (28)
  - General (53)
  - On-Premises (2)
  - POCs and Custom Solutions (16)
- News & Releases (25)
  - Conferences (13)
  - Product Releases (12)
- Template Design (7)
- Troubleshooting (12)
  - SSRS Reports (5)
- Video Library (20)
  - How-To Videos (8)
  - Product Showcases (6)
  - Recorded Webinars (8)

TAGS

Alerts Attachments AX 2012 R2 AX 2012 R3 Azure Blob storage Azure Files Batch Certificates Check Conference alert Configurable business documents Custom fonts D365FO Data entity DRA DSP class Electronic reporting Electronic signature Email distributor batch Emailing Email sending status Email templates Exchange Free Edition Full Edition Invoice MS Word OneBox PDF Performance Placeholders Print archive Print destinations Printing Print management Report parameters SharePoint SMTP SSRS Template design Version releases Video Viewer Workflow X++

Docentric AX Free Edition

We are proud to contribute to the community by delivering a really useful part of our software completely for free.

LEARN MORE

LATEST POSTS

- Docentric AX 3.4.4 Released
- Version 3.x Change Log
- Build Models, Sync DB and Deploy Reports Using CLI D365FO Tools
- Upgrade Azure Storage Emulator on D365FO OneBox Without Losing Your Files
- Designing Overflow Checks in D365FO
- Generate Sales Invoice Proforma as Byte Array
- DynamicsCon 2022: Print Reports from D365FO to Local Network via Azure Files
- Alerts++ in D365FO – Part 2
- Save Invoices to Local File System from D365FO in Cloud
- Save Invoices to Azure Files from D365FO
- Docentric AX 3.4.3 Released
- Run Email Distributor Batch for Different Outgoing Emails
- Batch email sending status improvements
- Change propagation between AOS nodes
- Notify Vendors That Certificates Are About to Expire Using Alerts in D365FO
- See all posts >>

AUTHORS

- Albin Lotric (25)
- Ana Gligorijevic (41)
- Boza Lotric (6)
- Goran Arsovski (2)
- Gregor Jovan (4)
- Irena Vujcic Pavlovic (7)
- Jernej Valic (16)
- Jovica Zivkovic (7)
- Jure Leskovec (4)
- Jure Suklje (1)
- Klemen Novak (14)
- Maja Kovac (11)
- Miha Vuk (11)
- Nenad Krantic (1)
- Sanja Kolundzija (28)
- Slobodan Vucicevic (4)
- Vilko Jenko (4)

Gold
Microsoft Partner
Microsoft

FOLLOW US

NEWSLETTER SUBSCRIBE

Your email (Required)

Back to top