DataCap SmartExport

(v*1.0*)

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Title** | **Description** | **Author(s)** |
| 02/Mar/2020 | MVP | Version 1.0. The first MVP release | DBA Asset Development team |

[Getting Started 3](#_Toc34057263)

[Overview 3](#_Toc34057264)

[Release Notes 3](#_Toc34057265)

[Installation 3](#_Toc34057266)

[Upgrade 3](#_Toc34057267)

[Dependencies 4](#_Toc34057268)

[Compatibility 4](#_Toc34057269)

[Common Tasks 4](#_Toc34057270)

[Template Development 4](#_Toc34057271)

[Template Structure 4](#_Toc34057272)

[Control Parameters 4](#_Toc34057273)

[Supported tags 5](#_Toc34057274)

[Sample templates and description 7](#_Toc34057275)

[Template Development tips 10](#_Toc34057281)

[Troubleshooting 13](#_Toc34057287)

[Logs 13](#_Toc34057288)

[Error Messages 13](#_Toc34057289)

[More Information 13](#_Toc34057290)

[Tech preview features 13](#_Toc34057291)

[Known Limitations 13](#_Toc34057292)

[Help and support 14](#_Toc34057293)

# Getting Started

## Overview

DataCap SmartExport is an action that when configured allows the DataCap developer to selectively output extracted data in a format acceptable by the downstream applications. The action can be associated at various levels of the DCO hierarchy depending on the need and is generally tied to the export step of the DataCap workflow. The action library contains a single method which accepts the path to the template file as input and generates one or more output file as dictated by the template. Detailed information on the template structure

## Release Notes

IBM® DataCap Custom Export Asset is now available to users for general or limited use. The asset includes a DataCap Export framework for enhanced data export ,from within DataCap, to enable easier processing by the downstream applications.

The DataCap export framework is an independent asset that fits into an overall business solution for scenarios where digital data extracted from documents need to be updated in external applications like SAP, Sales Force, ERPs etc. This framework will help DataCap developers to selectively export the most important data elements from documents, which is of business importance, and in a format of choice. Developers can use predefined templates for data formatting and provide conditional/looping controls to obtain the desired formatted content in the generated output.

## Installation

Follow the below steps for a fresh installation

* Close DataCap studio (and other DataCap development tools)
* Unzip the deployment bundle
* Copy the “SmartExportTemplates.dll” file from the unzipped folder into “<DataCap installed dir>/RSS” folder
* Start DataCap studio and find the SmartExportTemplates action library under global actions

## Upgrade

Follow the below steps for upgrade

* Backup your existing SmartExportTemplates.dll file
* Follow installation steps to deploy a new version of the “SmartExportTemplates.dll” file
* Start DataCap studio
* Modify parameters of the SmartExportTemplates action rulesets that complies to the new version. Read the release notes and product documentation.
* Freshly publish rulesets

## Dependencies

SmartExportTemplate action(s) are developed using the custom actions template supplied by IBM. In addition to the default library references in the custom template, SmartExportTemplate action also refers to “DataCap.Global” library which is expected to be in available in the default DataCap installation directory path.

## Compatibility

SmartExportTemplate action(s) are compatible with v9.1.6 version of DataCap. No other versions are tested against. Please contact support if there is a need to use this action library against a different version of DataCap.

# Common Tasks

## Template Development

Templates that are supplied to the SmartExportTemplate action are well formed XML file with a pre-defined syntax. Field selection, formatting and control statements are assembled using the respective tags. The action interprets the tags in its order from start to the end of the template file and writes output data in the same order. While technically it is possible to associate the action at any level of DCO and a workflow step of choice, the DataCap developer is expected to have the understanding of what he plans to generate and thereby supply a meaningful template and associate at the right level of the DCO tree and the appropriate workflow step. In most cases, the workflow step is the “export” step and the DCO level is decided based on the template.

## Template Structure

All supported tags are expected to be between the default start and end tags viz., <se:smartexport> and </se:smartexport> respectively. A set of control parameters allows the selection of filename and extension, locale etc. The tags are self-contained units which means a tag is evaluated on its whole and the expected output of such a tag is immediately written to the output buffer. There is no relation between preceding and succeeding base tags which are direct children of the <se:smartexport> tag. While using nesting, the inner loop/conditions operate on the context set by the outer control statements. Default namespace <https://www.w3.org/2001/XMLSchema> that comes with the sample template should be used as is.

## Control Parameters

The below control parameters are supported. If not supplied in the template, the default values are used.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tag name** | **Description** | **Default** | **Children** | **Mandatory** |
| se:appendToFile | If true a single output file is created for the entire batch and data appended to it every time the action is called | false |  | No |
| se:outputFolder | If specified, files generated by the action are written to this output folder. Default is used if error in path. | Batch directory | se:smartParam | No |
| se:locale | Holds the locale to be used for parsing the Date found in the input files. If not specified, the current system locale will be used. | System Locale | se:smartParam | No |
| se:filename | Used to specify the name of the output file that will be generated. If not specified a default name will be used. | SmartExport | se:smartParam | No |
| se:fileext | Used to specify file extension for the exported file. | txt |  | No |
| <se:memCacheLines> | Used to specify the maximum lines which is stored in memory. After the max lines is reached , data is to flushed to temporary file and written to actual file at the end of the export call. | 3000 |  | No |

## Supported tags

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tag name** | **Description** | **Attributes** | **Possible children** | **Base Tag** |
| se:smartexport | The root node of the template. Every other export nodes should be assembled within this root node. | Version and namespace | All base tags | N/A |
| se:data | Contains the content to be printed in the output file, it could be plain text or a reference to a field in the DCO that can be specified in the se:value tag. Additional formatting and smart parameter tags are supported as well. |  | se:value  se:tab  se:comma  se:smartParam  text | Yes |
| se:value | Holds the reference to a field in the DCO whose value would be printed in the output file. | select=<DCO reference> |  | No |
| se:tab | Used to print tabs in the output file. |  |  | No |
| se:comma | Used to print comma in the output file. |  |  | No |
| se:if | Conditional control. If “test” evaluates to true, child tags are evaluated  **Note**: Technical preview features allows for nesting. | test=<condition> | se:elsif  se:else  se:data  se:if  se:for-each | Yes |
| se:elsif | Child of “if” for multi condition checks. | test= <condition> | se:data  se:if  se:for-each | No |
| se:else | Child of “if” for default output when all conditions fail. |  | se:data  se:if  se:for-each | No |
| se:for-each | Looping control. Context is based on the current DCO to which the action is associated. Works on current DCO’s children.  **Note**: Technical preview features allow for nesting whereby grandchildren of the current DCO can be looped through.  **Example**: To loop through the Documents in a batch, action should be associated at a batch level and the value for select would be “document”. Similarly, for iterating over pages the association would be at document level and select value would be “page”. | select=<DCO level to be looped through> | se:if  se:data  se:for-each | Yes |
| se:smartParam | Used to specify the smart parameter. Smart parameters are allowed in control and data tags. Currently in control tags, smart parameters cannot be combined/concatenated with static text.  **Example**: <se:smartParam>BATCHID</se:smartParam>  **Note**: This is a technical preview feature |  |  | No |

## Sample templates and description

Below are some of the sample templates and their description. Sample templates refer to the DCO in the “TravelDocs” application and it is meant to showcase the features of the SmartExportTemplates action.

### Field level

Associated to the Airfare field of AirTicket page of TravelDocs. Outputs the value of Airfare and for a given batch writes all the output into a single file

<se:smartexport version="1.0" xmlns:se="https://www.w3.org/2001/XMLSchema">

<se:appendToFile>true</se:appendToFile>

<se:locale>en-US</se:locale>

<se:filename>FIELD\_AirfareList</se:filename>

<se:fileext>txt</se:fileext>

<se:outputFolder>c:\SmartExportOutput</se:outputFolder>

<se:data>

<se:value select="[DCO].[Flight].[Air\_Ticket].[Airfare]"/>

</se:data>

</se:smartexport>

### Page level

Associated at Air\_Ticket level of TravelDocs. Generates a single file (see appendToFile is **true**) with tab delimited values for each of the page processed

<se:smartexport version="1.0" xmlns:se="https://www.w3.org/2001/XMLSchema">

<se:appendToFile>true</se:appendToFile>

<se:locale>en-US</se:locale>

<se:filename>PAGE\_AllFlightDetails</se:filename>

<se:fileext>csv</se:fileext>

<se:outputFolder>c:\SmartExportOutput</se:outputFolder>

<se:data>

<se:imagename/><se:tab/>

<se:value select="[DCO].[Flight].[Air\_Ticket].[Return\_From]"/><se:tab/>

<se:value select="[DCO].[Flight].[Air\_Ticket].[Return\_To]"/><se:tab/>

<se:value select="[DCO].[Flight].[Air\_Ticket].[Airfare]"/><se:tab/>

<se:value select="[DCO].[Flight].[Air\_Ticket].[Total\_Cost]"/>

</se:data>

</se:smartexport>

### Document level

Associated at Flight level of TravelDocs. Generates one file each for the processed document. Generates key/value pair as output

<se:smartexport version="1.0" xmlns:se="https://www.w3.org/2001/XMLSchema">

<se:appendToFile>false</se:appendToFile>

<se:locale>en-US</se:locale>

<se:filename>DOCUMENT\_FlightDetails</se:filename>

<se:fileext>txt</se:fileext>

<se:outputFolder>c:\SmartExportOutput</se:outputFolder>

<se:if test="[DCO].[Flight].[Air\_Ticket].[Return\_Date] GREATER-THAN 01/Aug/2010">

<se:data>Return From = <se:value select="[DCO].[Flight].[Air\_Ticket].[Return\_From]"/></se:data>

<se:data>Return To = <se:value select="[DCO].[Flight].[Air\_Ticket].[Return\_To]"/></se:data>

<se:data>Airfare = <se:value select="[DCO].[Flight].[Air\_Ticket].[Airfare]"/></se:data>

<se:data>Total Cost = <se:value select="[DCO].[Flight].[Air\_Ticket].[Total\_Cost]"/></se:data>

</se:if>

</se:smartexport>

### Batch level

Attached at the batch level of TravelDocs. Generates a CSV delimited file for car rentals.

<se:smartexport version="1.0" xmlns:se="https://www.w3.org/2001/XMLSchema">

<se:appendToFile>false</se:appendToFile>

<se:locale>en-US</se:locale>

<se:filename>BATCH\_Comma\_Separated\_Car\_Rental</se:filename>

<se:fileext>csv</se:fileext>

<se:outputFolder>c:\SmartExportOutput</se:outputFolder>

<se:data>Pickup\_Date<se:comma/>Pickup\_Location<se:comma/>Return\_Date<se:comma/>Return\_Location<se:comma/>TotalCost</se:data>

<se:for-each select="document">

<se:if test="document.type EQUALS Car\_Rental ">

<se:data>

<se:value select="[DCO].[Car\_Rental].[Rental\_Agreement].[Pickup\_Date]"/><se:comma/>

<se:value select="[DCO].[Car\_Rental].[Rental\_Agreement].[Pickup\_Location]"/><se:comma/>

<se:value select="[DCO].[Car\_Rental].[Rental\_Agreement].[Return\_Date]"/><se:comma/>

<se:value select="[DCO].[Car\_Rental].[Rental\_Agreement].[Return\_Location]"/><se:comma/>

<se:value select="[DCO].[Car\_Rental].[Rental\_Agreement].[Total\_Cost]"/>

</se:data>

</se:if>

</se:for-each>

</se:smartexport>

### Batch level with Nesting (Technical preview)

Attached at a batch level of TravelDocs. This template is to demostrate the technical preview features of nesting and smart parameters.

<se:smartexport version="1.0" xmlns:se="https://www.w3.org/2001/XMLSchema">

<se:appendToFile>false</se:appendToFile>

<se:locale>en-US</se:locale>

<se:filename>

<se:smartParam>BATCHID</se:smartParam>

</se:filename>

<se:fileext>txt</se:fileext>

<se:outputFolder>c:\SmartExportOutput</se:outputFolder>

<se:for-each select="document">

<se:for-each select="page">

<se:if test="page.type EQUALS Air\_Ticket">

<se:data>Page ID: <se:value select="page.name"/></se:data>

<se:data>Return Date: <se:value select="[DCO].[Flight].[Air\_Ticket].[Return\_Date]"/></se:data>

<se:if test="[DCO].[Flight].[Air\_Ticket].[Return\_Date] GREATER-THAN 01/Aug/2010">

<se:data><se:tab/>Return From : <se:value select="[DCO].[Flight].[Air\_Ticket].[Return\_From]"/></se:data>

<se:data><se:tab/>Return To : <se:value select="[DCO].[Flight].[Air\_Ticket].[Return\_To]"/></se:data>

<se:data><se:tab/>Airfare : <se:value select="[DCO].[Flight].[Air\_Ticket].[Airfare]"/></se:data>

<se:data><se:tab/>Total Cost : <se:value select="[DCO].[Flight].[Air\_Ticket].[Total\_Cost]"/></se:data>

<se:elsif test="[DCO].[Flight].[Air\_Ticket].[Return\_Date] LESSER-THAN 01/Aug/2010">

<se:if test="[DCO].[Flight].[Air\_Ticket].[Return\_Date] GREATER-THAN 01/Jan/2005">

<se:data><se:tab/>Airfare : <se:value select="[DCO].[Flight].[Air\_Ticket].[Airfare]"/></se:data>

<se:data><se:tab/>Total Cost : <se:value select="[DCO].[Flight].[Air\_Ticket].[Total\_Cost]"/></se:data>

</se:if>

</se:elsif>

</se:if>

</se:if>

</se:for-each>

</se:for-each>

</se:smartexport>

## Template Development tips

### Field References

The DCO field references should be of the format [DCO].[Document\_Name].[Page\_Name].[Field\_Name]. Whenever the smart export processor encounters such an expression it would try to fetch the value for the same from the current DCO and replace it for either evaluating a condition within the IF block or just printing the value when used within the DATA block.

For example, if the document hierarchy was as shown in the image below, and we want to refer to the field Total\_Cost, the DCO expression would be [DCO].[Car\_Rental].[Rental\_Agreement].[Total\_Cost].

A screenshot of a cell phone

Description automatically generated

### DCO type expressions

These are used to check the type of a document or page. Whenever the smart export processor encounters such an expression it would try to fetch the value for the same from the current DCO and replace it for evaluating a condition within the IF block.

The expressions that are currently supported are as follows:

document.type

page.type

Considering the examples used in the document, the values for document.type could be Car\_Rental and page.type could be Rental\_Agreement

### Conditions

The value specified against the test attribute of the se:if/se:elsif tags are referred to as conditions. Conditions can hold a simple expression or complex expressions concatenated by the and/or keywords. These expressions would be evaluated to provide a Boolean result. The conditions can contain DCO references or literal values. The data types supported are String, DateTime, Boolean, Integer and Floating-point numbers. When the DCO references are replaced with their actual values, they should fall under the supported data types only. The operators supported are EQUALS, GREATER-THAN and LESSER-THAN.

<se:if test="document.type EQUALS Car\_Rental ">

<se:data>Doc type is car rental</se:data>

<se:data> Total\_Cost = <se:value select="[DCO].[Car\_Rental].[Rental\_Agreement].[Total\_Cost]"/></se:data>

<se:elsif test="document.type EQUALS Flight and [DCO].[Flight].[Air\_Ticket].[Airfare] LESSER-THAN 10000 ">

<se:data>Doc type is flight and airfare is lesser than 10000</se:data>

<se:data> Airfare = <se:value select="[DCO].[Flight].[Air\_Ticket].[Airfare]"/></se:data>

</se:elsif>

<se:elsif test="[DCO].[Flight].[Air\_Ticket].[Airfare] GREATER-THAN 10000">

<se:data>Airfare is greater than 10000</se:data>

<se:data> Airfare =<se:value select="[DCO].[Flight].[Air\_Ticket].[Airfare]"/></se:data>

</se:elsif>

<se:else>

<se:data>Default</se:data>

<se:data> Airfare =<se:value select="[DCO].[Flight].[Air\_Ticket].[Airfare]"/></se:data>

</se:else>

</se:if>

### Tag Nesting

The sample below illustrates the usage of **FOR-EACH within IF**

<se:if test="document.type EQUALS Flight">

<se:for-each select="page">

<se:data>Doc type is Flight</se:data>

<se:if test="page.type EQUALS Insurance">

<se:data>Page type is Insurance</se:data>

<se:else >

<se:if test="page.type EQUALS Air\_Ticket">

<se:data>Page type is Air\_Ticket</se:data>

<se:if test="[DCO].[Flight].[Air\_Ticket].[Airfare] LESSER-THAN 10000">

<se:data>Airfare is lesser than 10000</se:data>

<se:data>Airfare=<se:value select="[DCO].[Flight].[Air\_Ticket].[Airfare]"/></se:data>

</se:if>

</se:if>

</se:else>

</se:if>

</se:for-each>

</se:if>

Sample below illustrates **IF within IF**

<se:if test="document.type EQUALS Car\_Rental">

<se:data>Doc type is Car\_Rental</se:data>

<se:elsif test="document.type EQUALS Flight">

<se:data>Doc type is Flight</se:data>

<se:if test="page.type EQUALS Air\_Ticket">

<se:data>Page type is Air\_Ticket</se:data>

<se:if test="[DCO].[Flight].[Air\_Ticket].[Airfare] LESSER-THAN 10000">

<se:data>Airfare is lesser than 10000</se:data>

<se:data>Airfare=<se:value select="[DCO].[Flight].[Air\_Ticket].[Airfare]"/></se:data>

</se:if>

</se:if>

</se:elsif>

</se:if>

Sample below illustrates FOR**-EACH within FOR-EACH**

<se:for-each select="document">

<se:for-each select="page">

<se:if test="page.type EQUALS Air\_Ticket">

<se:data>Page ID: <se:value select="page.name"/></se:data>

<se:data><se:tab/>Return From : <se:value select="[DCO].[Flight].[Air\_Ticket].[Return\_From]"/></se:data>

<se:data><se:tab/>Return To : <se:value select="[DCO].[Flight].[Air\_Ticket].[Return\_To]"/></se:data>

<se:data><se:tab/>Airfare : <se:value select="[DCO].[Flight].[Air\_Ticket].[Airfare]"/></se:data>

<se:data><se:tab/>Total Cost : <se:value select="[DCO].[Flight].[Air\_Ticket].[Total\_Cost]"/></se:data>

</se:if>

</se:for-each>

</se:for-each>

### Smart parameters

**Note**: This is a technical preview feature.

While using Smart parameters, DataCap developer is expected to be aware of the context where a particular Smart parameter yields its value. This depends on the level of the DCO where the action is associated with. For example, it the batch ID is to be used as the filename while associating the action at the batch level, below snippet can be used for the filename parameter

<se:filename>

<se:smartParam>ID</se:smartParam>

</se:filename>

# Troubleshooting

## Logs

Log messages are written to the default log files. For example, if the action is associated with the export step, find the logs in the “export\_rrs.log” file. Log levels are controlled using default DataCap administration tools and the SmartExportTemplate action abides by it.

## Error Messages

Search the log files for the text “SmartExport”. Most of the logs from the action are written with the prefix to help spot the logs during debugging. Along with the prefix, the level related information will also help identifying errors/exceptions during execution. Info messages can generally be ignored.

# More Information

## Tech preview features

Nesting and Smart Parameters are technical preview features. Sanity checks are done for these features but extensive testing with various possible use cases are not done. It is recommended to use these with caution

## Known Limitations

In the current release the following options are not supported-

1. Support for complex types like table data
2. Support for smart parameters along with plain text

## Help and support

Contact the DBA Asset development team for help and support.

Offering Manager: Poornima Sekhar (psekhar0@in.ibm.com)

Development Manager: Shashank Gopalakrishna (shagopal@in.ibm.com)

Technical Architect: Ranjeeth Pasupathi (rpasupat@in.ibm.com)