Version: 2.18.X

Advanced Mapping Guide

Sample Files

Two sample Advanced Collection Builder `.docx` templates can be found at the following links. Each contains various examples of how a Word template can be formatted to obtain a highly customized collection report export.

- Sample File 1 shows all the major fields and formats exportable from a collection.
- Sample File 2 shows how to work with specific helpers to target the export of specific custom fields, events, and data fields.

Syntax Overview

Several basic data fields are easily referenced in the `.docx` file. Refer to a collection data field in your template by wrapping a data field name with the following syntax:

```
+++= data-field +++
```

> **INFO**
>
> Include `+++= title +++` in your file to export the `title` of the collection.

Fields with Additional Options

Some data fields have multiple output options to specify. The syntax for these fields is:

```
+++= data-field.option
```

> **TIP**
>
> To get the `createdAt` time of a collection, but in local time format, include `+++= createdAt.dateTimeLocal +++` in your document.

Working with Loops

Certain data fields in a collection contain multiple elements (such as Participants, Comments, Events). You can loop through any of these elements and selectively output data.

The syntax for a loop is structured as follows. You can name the variable `X` whatever you like:

> **TIP**

```
+++FOR X IN data_field_that_is_loopable+++    // Starts the loop.

+++= $X.key+++                               // Output the data you
desire.

+++END-FOR X+++                              // Ends the loop.
```

When using a FOR-loop, do not forget to **include the `$` sign** on the reference item.

> **TIP**
>
> Assume we are attempting to loop through the participants in a collection and desire to output each participant's full name and email in a bulleted list. This would look like the following:

```
+++FOR person IN participants +++

   - +++= $person.fullName +++
   - +++= $person.email +++

+++END-FOR person +++
```

Utilizing helper functions and `conditionals` can also help output specific information that you define (e.g., specific custom fields, event types, or event fields).

Basic Fields

Basic fields are easily mapped using the following syntax:

Data Field	Syntax
title	`+++= title +++`
description	`+++= description +++`
numItems	`+++= numItems +++`
priority	`+++= priority +++`
id	`+++= id +++`

Tags

Tags include some fields with additional options, and can be mapped using the following syntax:

Data Field	Loop Required?	Syntax
primaryTag	-	`+++= primaryTag.tagOption +++`
additionalTags	Yes	`+++= additionalTags.tagOption +++`
additionalTagsList	-	`+++= additionalTagsList +++`

When using `primaryTag` or `additonalTags`, you must specify a tag option. Replace `tagOption` with any of the following:

Option	Output Example
id	3741e133-d830-4931-845e-49333b4f519d
name	LE/CI - Complete
color	#37e04d
order	15
type	SYSTEM

TIP

To get the primary tag's name:

`+++= primaryTag.name +++`

To loop through all additional tags and output the tag name:

```
+++FOR tag IN additionalTags +++

+++= $tag.name +++

+++END-FOR tag+++
```

Users

Users include some fields with additional options, and can be mapped using the following syntax:

Data Field	Loop Required?	Syntax
createdByUser	No	`+++= createdByUser.userOption +++`

Data Field	Loop Required?	Syntax
assignedToUser	No	`+++= assignedToUser.userOption +++`
participants	Yes	`+++= participants +++`

When using `createdByUser`, `assignedToUser`, or `participants`, you must specify a user option. Replace `userOption` with any of the following:

Option	Output Example	Notes
id	70d3d3a6-8c52-4483-8610-e4da9192775c	N/A
firstName	John	N/A
lastName	Doe	N/A
email	`jdoe@cogility.com``	N/A
role	Threat Analyst	N/A
fullName	John Doe	N/A
fullNameAndEmail	John Doe (`jdoe@cogility.com``)	N/A
fullNameAndRole	John Doe (Threat Analyst)	N/A
groups	Group One, Group Two	N/A
createdAt.*option*	`DateTime`	See Datetime Fields
deletedAt.*option*	`DateTime`	See Datetime Fields
updatedAt.*option*	`DateTime`	See Datetime Fields

Option	Output Example	Notes
lastLoginAt.*option*	`DateTime`	See Datetime Fields

TIP

Get the name of a collection's assigned user and their email in two separate lines.

```
+++= assignedToUser.fullName +++

+++= assignedToUser.email +++
```

Loop through all participants and output their full name and last login date in local time (at the time the export was performed):

```
+++FOR person IN participants +++

+++= $person.fullName +++

+++= $person.lastLoginAt.dateTimeLocal +++

+++END-FOR person+++
```

Datetime Fields

Datetime fields include some fields with additional options, and can be mapped using the following syntax:

Data Field	Loop Required?	Syntax
archivedAt	No	`+++= archivedAt.dateTimeOption +++`
createdAt	No	`+++= createdAt.dateTimeOption +++`

Data Field	Loop Required?	Syntax
updatedAt	No	`+++= updatedAt.dateTimeOption +++`
deletedAt	No	`+++= deletedAt.dateTimeOption +++`

When using a time-based field like `archivedAt`, `createdAt`, `updatedAt`, or `deletedAt`, you must specify a datetime option to output a specific datetime format. Replace `dateTimeOption` with any of the following:

Option	Output Example
dateTimeLocal	October 15th, 2022 at 11:06 AM
dateTimeLocalTZ	October 15th, 2022 at 11:06 AM PDT
dateTimeUTC	October 15th, 2022 at 6:06 PM UTC
dateLocal	October 15th, 2022
dateLocalTZ	October 15th, 2022 PDT
dateUTC	October 15th, 2022 UTC
shortLocal	10/15/2022
shortLocalTZ	10/15/2022 PDT
shortUTC	10/15/2022 UTC
isoDate	2022-10-15T18:06:55Z
shortLocalTZ	10/15/2022 PDT
shortUTC	10/15/2022 UTC

Option	Output Example
isoDate	2022-10-15T18:06:55Z
timeLocal	11:06 AM
timeLocalTZ	11:06 AM PDT
timeUTC	11:06 AM UTC

TIP

Output a collection's createdAt time in date time local format:

```
+++= createdAt.dateTimeLocal +++
```

Custom Fields

Users have the option of:

- Picking specific custom fields to export by referring to a Custom Field's `key` (found in Workstation).
- Exporting all Custom Fields in a collection.

Regardless of the method used, the following custom field options must be specified to output a specific value.

Custom Field Options

Each Custom Field in Workstation contains the following options that can be output:

Option	Output Example	Notes

Option	Output Example	Notes
id	34b830b0-ee6b-4646-87a3-cf0d2925fb41	N/A
key	toohzx	N/A
type	CHECKBOX	N/A
name	Type of Case	N/A
value	Domestic Violence, Burglary	N/A
values	---	Helpful for looping through checkboxes and multichips for custom formatting.

Referencing Specific Custom Fields

Custom fields do not require a loop if you do not need specific control of the output format.

To reference a custom field:

1. Find the key for the desired custom field template in **Designer** > **Custom Field Templates** in the edit mode for the template.
2. Reference this key in your doc template with the syntax: `+++= customFieldByKey.keyhere.option +++`, where `option`` is any of the available custom field options.

TIP

Assume there is a `Subject SSN` custom field with a key of `ABCDEF`. To output the name and value of this field:

```
+++= customFieldByKey.ABCDEF.name +++   // Outputs 'Subject SSN'

+++= customFieldByKey.ABCDEF.value +++  // Outputs the value
```

NOTE

Custom fields of the checkbox or multichip type output multiple selections as a concatenated string separated by commas. A loop is required for more advanced control over data outputs (such as bulleted or numbered lists of checkbox values).

Rich Text Fields

When exporting rich text custom fields, the tool attempts to maintain the format of the rich text content and styling. However, there may be situations in which Microsoft Word overrides specific formatting, such as indentation levels within bulleted and numbered lists or section headers.

Additionally, it is generally recommended to use the _html helper to ensure that the output of the rich text content closely resembles its appears in Workstation.

TIP

Output a rich text field using the HTML rendering method (defaults to Arial font and 12pt font size):

```
+++HTML _html(customFieldByKey.keyhere.value) +++
```

Looping through Checkboxes and Multichip Fields

Checkbox and Multichip custom fields may possess multiple selected options. Looping through either field requires the `values` option instead of `value`.

TIP

Assume there is a `Case Category` Custom Field with a key of `ABCDE`. To output the name of this field and each applied value as a numbered list:

```
+++= customFieldByKey.ABCDE.name +++          // Outputs the name "Case
Category".

+++ FOR val in customFieldByKey.GHIJK.values  // Starts the loop.

1. +++= $val +++                              // Outputs the value.

+++END-FOR val+++                             // Closes the loop.
```

Exporting All Custom Fields

If desired, you may choose to output all custom fields in a collection. This is achieved by doing a loop on the `customFields` data field.

TIP

Loop through all custom fields and output their names and values:

```
+++ FOR field in customFields +++             // Starts the loop.

+++= $field.name +++                          // Outputs the field name.

+++HTML _html($field.value) +++               // Outputs the field value
and parses HTML (for rich text fields).

+++ END-FOR field +++                         // Closes the loop.
```

If needed, the value of the custom field can also be wrapped in the `HTML` renderer or the `HTML Helper`. This is useful for handling rich text custom fields.

Items (Events)

Items are data elements in a collection that represent the events added to a collection. Items can **only be accessed with a FOR-loop**.

Data Field	Output Example	Notes
eventTypeTitle	Behavior Profiles	N/A
coreId	021a773e-de6e-2029-0a9d-5d528022065a	N/A
primaryTitle	John Doe	N/A
createdAt	August 7th, 2023 at 11:54 AM	N/A
updatedAt	October 24th, 2023 at 11:10 AM	N/A
riskScore	91	N/A
lexicons	compare, financial, bank	N/A
eventDetails	---	Loop within the Items loop. See Event Detail Options
eventTypeTopic	behavior_profiles	N/A
eventTypeProjectName	Project Name Here	N/A
eventTypeDataSource.id	021a773e-de6e-2029-0a9d-5d528022065a	N/A
eventTypeDataSource.connectString	kafka.confluent.svc.cluster	N/A

Data Field	Output Example	Notes
eventTypeDataSouce.name	default	N/A
eventTypeDataSource.type	kafka	N/A

TIP

Loop through all attached events in a collection and output the event type name, then display the core ID and risk score as a bulleted list:

```
+++ FOR event in items +++           // Starts the loop.

 +++= $event.eventTypeTitle +++      // Outputs the event name.
- +++= $event.coreId +++             // Outputs core ID as a bullet point.
- +++= $event.riskScore +++          // Outputs risk score as a bullet
point.

+++END-FOR event +++                 // Closes the loop.
```

Event Detail Options

The `eventDetails` data fields on Items are also loops. You can access these loop while looping through `Items` and output any event's detailed fields.

Event Detail Field	Output
type	string
path	string
name	string
value	string

Event Detail Field	Output
key	string

TIP

Loop through all attached events on a collection and output the event type name, then each event detail field's name and value as an indented list:

```
+++ FOR event in items +++                          // Starts the items loop.

+++= $event.eventTypeTitle +++

+++ FOR field in $event.eventDetails +++            // Starts a second loop
in the event details.

- +++= $field.name +++: +++= $field.value +++       // Outputs the bolded
field name and value as a bullet point.

+++ END-FOR field +++                               // Closes the second
event details loop.

+++END-FOR event +++                                // Closes the items loop.
```

Outputting Specific Event Types and Fields

It is possible to identify specific event types to output as well as specific fields in each type of event. This is more advanced, but is possible through several helper functions with a FOR-loop:

1. Find the key for the desired Event Detail Field.
   i. Navigate to the **Admin** page of Workstation.
   ii. Click the **Options** button for the desired Event Type and select **View Event Definition**.
   iii. The keys for various data fields and event metadata are shown.
2. Reference this key in your doc template with the syntax: `+++= _get($event, 'eventDetailByKey.keyhere.option') +++`:
   - Where `option` is any of the available event detail options.

- The _get helper is highly recommended for this use case.

> **TIP**
>
> Assume a collection has numerous types of events added, and you desire to:
>
> - Only export the `Behavior Profiles` events.
> - Only output the `SSN` (key of `ABCD`) and `Description` (key of `EFGH`) fields.
> - Manually show the field names `Social Security Number` and `Short Description`.

```
// Start the loop and only keep Behavior Profile events.

+++FOR events IN _keep(items, ‘eventTypeTitle’, ‘Behavior Profiles’)+++

Social Security Number

++++= _get($events, 'eventDetailByKey.ABCD.value') +++     // Get the SSN
data field.

Short Description

++++= _get($events, 'eventDetailByKey.EFGH.value') +++     // Get the
Description data field.

+++END-FOR events +++                                     // Close the
loop.
```

Comments

Comments on a collection can **only be accessed with a FOR-loop**.

Option	Output	Syntax
id	c940f633-2158-487a-a7db-bbab57b7de6e	`+++= $comment.id +++`

Option	Output	Syntax
contentHTML	---	`+++= $comment.contentHTML +++`
contentPlain	---	`+++= $comment.contentPlain +++`
createdAt	See Datetime Fields	`+++= $comment.createdAt.dateTimeOption +++`
deletedAt	See Datetime Fields	`+++= $comment.deletedAt.dateTimeOption +++`
updatedAt	See Datetime Fields	`+++= $comment.updatedAt.dateTimeOption +++`
user	See Users	`+++= $comment.user.userOption +++`
replies	Loop-able within comments	See example below

TIP

Output each comment on the collection and include:

- The user's full name and email.
- The date and time the comment was created.
- The comment rendered in plain text format.
- If there is a reply to a comment, output the:
  - Full name of the reply author.
  - Date and time the reply was made.
  - The content of the reply rendered in plain text format.

A

The
col

_n

A c
tim
foll

++

```
+++ FOR comment in comments +++                          // Start the
comments loop.

By: +++= $comment.user.fullNameAndEmail +++              // Output name and
email of commenter.

Created: +++= $comment.createdAt.dateTimeLocalTZ +++     // Output "Created
At" time.

+++= $comment.contentPlain +++                           // Output comment
content in plain text.

+++ FOR reply in $comment.replies +++                    // Start loop for
replies

    +++= $reply.user.fullName +++                        // Output name of
reply author

    +++= $reply.createdAt.dateTimeLocalTZ +++            // Output date/time
reply was made

    +++= $reply.contentPlain +++                         // Output reply
content in plain text.

++ END-FOR reply +++                                     // Close the reply
loop.

+++END-FOR comment+++                                    // Close the
comments loop.
```

- `Desired Font` is optional. The default is `Arial`.
- `Desired Font Size` is optional. The default is `12pt`.

> **TIP**
>
> Assume there is a rich text custom field with a key of `ABCDE`. To output the name of this field and the value in HTML format:
>
> `+++HTML _html(customFieldByKey.ABCDE.value) +++`

_plain

Exports a value as plain text. Using this with custom fields that are rich text or comments may export and show HTML tags and markdown characters. The syntax is as follows:

```
+++= _plain(value) +++
```

> **TIP**
>
> Assume there is a rich text custom field with a key of `ABCDE`. To output the name of this field and the value in plain text format:
>
> ```
> +++= _plain(customFieldByKey.ABCDE.value) +++
> ```

_get

The `_get` helper is useful, in combination with the **FOR-loop**, to grab a specific value with built-in error handling for null or non-existent values. If desired, you can specify the value to return if no value is found. The syntax (within a for-loop) is as follows:

```
+++= _get($event, 'eventDetailByKey.keyhere.option', Desired Null Value) +++
```

The `Desired Null Value` is optional. By default, null values return as `N/A`.

> **TIP**
>
> Assume a collection has numerous types of events added, and you desire to:
>
> - Only export the `Behavior Profiles` events.
> - Only output the `SSN` (key of `ABCD`) field.
> - Manually show the field name as `Social Security Number` in **bold**.
>
> ```
> +++FOR events IN _keep(items, ‘eventTypeTitle’, ‘Behavior Profiles’)+++
> // Start the loop.
>
> Social Security Number
> ```

```
+++= _get($events, 'eventDetailByKey.ABCD.value') +++
// Get the SSN data field.

+++END-FOR events +++
// Close the loop.
```

_omit

Filters out items in a list by comparing the values of the given key. **Used in a FOR-loop**.

The syntax is:

`_omit(list, ID, id1, id2, ...)`

Where:

- `list` is a filterable list (e.g., `items`, `eventDetails`, `customFields`).
- `ID` is the field you wish to filter by.
- `id1`, `id2`, etc. are the values to filter by.

TIP

Loop through the events (items) in a collection and **exclude** any event that is under "Behavior Profiles":

```
+++FOR events IN _omit(items, ‘eventTypeTitle’, ‘Behavior Profiles’)+++
// Start the loop.

Social Security Number

+++= _get($events, 'eventDetailByKey.ABCD.value') +++
// Get the SSN data field.

Short Description

+++= _get($events, 'eventDetailByKey.EFGH.value') +++
// Get the Description data field.
```

```
+++END-FOR events +++
// Close the loop.
```

_keep

Keeps only the items in a list by matching the values of the given key. **Used in a FOR-loop**.

The syntax of the `_keep` helper is as follows:

`_keep(list, ID, id1, id2, ...)`

Where:

- `list` is a filterable list (e.g., `items`, `eventDetails`, `customFields`).
- `ID` is the field you wish to filter by.
- `id1`, `id2`, etc. are the values to filter by.

TIP

Loop through the events (items) on a collection and **only include** any event that is under "Behavior Profiles":

```
+++FOR events IN _keep(items, ‘eventTypeTitle’, ‘Behavior Profiles’)+++
// Start the loop.

Social Security Number

+++= _get($events, 'eventDetailByKey.ABCD.value') +++
// Get the SSN data field.

Short Description

+++= _get($events, 'eventDetailByKey.EFGH.value') +++
// Get the Description data field.

+++END-FOR events +++
// Close the loop.
```

Tags: report builder advanced DOCX report