Events after creating a SQUEEZE document (CreateSource)

Audience

This documentation is intended for third-party developers who want to extend the **creation process of a SQUEEZE document** (tables `DXP SQZ Document Header`/`DXP SQZ Document Line`).

The focus is on the **integration events** that are raised *after* or *during* `ISourceDocument.CreateSource(...)`.

Context: where is `CreateSource` called?

The entry point for the SQUEEZE integration is in the **`DXP SQZ API Mgt.`** codeunit:

- A Core document (`DXP Document`) is created (via `DXP Document Mgt.`).
- The **`DXP ISource Document`** interface is then resolved via the document class and `CreateSource(Document)` is executed.

Simplified flow (from `CreateCoreAndSQUEEZEDocument`):

1. `DocumentMgt.AddDocument(DocClass, ...)` (create the Core document)
2. `ISourceDocument.CreateSource(Document)` (create the SQUEEZE document)
3. Process the import queue / "finish" the SQUEEZE document

Document classes & default behavior

Which `CreateSource` implementation is executed is defined in the enum extension **`DXP Document Class Ext.`**.

- **`DXP Invoice / Credit Memo`** → codeunit **`DXP SQZ P. Inv/Crdt Memo Impl.`**
- **`DXP Order Confirmation`** → codeunit **`DXP SQZ P. Order Conf. Impl.`**

Default (when no specific implementation is set): The `DXP Document Class` enum defines a `DefaultImplementation` on **`DXP Source Document Def. Impl.`**. This default implementation:

- does **not** create a SQUEEZE document,
- `CreateSource(...)` returns **`false`**.

Common pipeline inside `CreateSource`

Both SQUEEZE implementations follow (in short) this pattern:

1. Guard: the Core document must have `Status = Imported` and `JSON Raw` must exist.
2. `DXP SQZ Document Mgt.`:
   - `CreateSQUEEZEDocHeader(Document, DocHeader, RawJson)`
   - `CreateSQUEEZEDocLine(DocHeader, RawJson)`
3. `DXP Document Mgt.`: `UpdateDocumentAfterTransfer(...)` (set the Core document to "Transferred" and link it to the SQZ header)
4. Order match / autocomplete / attachments / validation (depending on document class and setup)

The most important events are therefore attached to:

- the header/line creation in `DXP SQZ Document Mgt.`
- the completion of the `CreateSource` procedure in the respective implementations
- optional steps such as order match or validation

Event matrix (short overview)

Area	Event	Timing	Control	Typical use cases
`DXP SQZ API Mgt.`	`OnAfterInitMapping(...)`	after initializing the mapping/token context	–	extend/check mapping
`DXP SQZ API Mgt.`	`OnAfterSaveAttachment(...)`	after an attachment has been saved successfully	–	post-processing/indexing
`DXP SQZ API Mgt.`	`OnAfterSetOriginFileNameOnBeforeModifyDocumentAttachment(...)`	before the attachment record is modified	–	influence file name/origin
`DXP SQZ P. Inv/Crdt Memo Impl.`	`OnAfterCreateDocLines`	directly after `CreateSQUEEZEDocLine(...)`	–	evaluate lines/RawJson after creation
`DXP SQZ P. Inv/Crdt Memo Impl.`	`OnAfterCreateSource`	at the end of `CreateSource(...)`	–	post actions after autocomplete/order match/validation

Area	Event	Timing	Control	Typical use cases
`DXP SQZ P. Inv/Crdt Memo Impl.`	`OnBeforeDoPlausibilityChecks`	before the plausibility checks	`IsHandled`	replace/extend checks
`DXP SQZ P. Inv/Crdt Memo Impl.`	`OnAfterDoPlausibilityChecks`	after the plausibility checks	–	reuse the result (entries)
`DXP SQZ P. Order Conf. Impl.`	`OnBeforeDoPlausibilityChecks`	before the plausibility checks	`IsHandled`	replace/extend checks
`DXP SQZ Document Mgt.`	`OnBeforeModifySQUEEZEDocumentHeader`	in `CreateSQUEEZEDocHeader(...)` before the final modify	–	add/normalize header values from `RawJson`
`DXP SQZ Document Mgt.`	`OnBeforeTransferRecognizedValuesOnAfterAssignRecognizedValues`	in standard line post-processing before `TransferRecognizedValues`	–	correct recognized values per line
`DXP SQZ Document Mgt.`	`OnAfterTransferRecognizedValuesOnBeforeInsertDocLine`	in standard line post-processing after `TransferRecognizedValues`	–	line-based derivations (account assignment/dimensions)
`DXP SQZ Document Mgt.`	`OnBeforePerformAutomaticOrdermatch`	before the automatic order match	`IsHandled`	take over order match completely/adjust OrderNoList
JSON creation	`OnBeforeAddLineToDocumentJObj`	before the line structure is built	–	extend the header JSON
JSON creation	`OnBeforeAddLineJObjToLineJArray`	before `LineJArray.Add(LineJObj)`	–	extend the line JSON

Events in `DXP SQZ P. Inv/Crdt Memo Impl.` (document class: Invoice / Credit Memo)

This codeunit provides the **strongest extension points directly in the `CreateSource` flow**.

`OnAfterCreateDocLines`

Raised directly after `CreateSQUEEZEDocLine(...)` (that is, once the header & lines exist).

Signature:

```
[IntegrationEvent(false, false)]
local procedure OnAfterCreateDocLines(DocHeader: Record "DXP SQZ Document Header"; RawJson: JsonObject)
```

Typical use cases:

- additional header initialization (for example custom flags/fields)
- additional line-based evaluations directly after creation
- extract custom metadata from `RawJson`

`OnAfterCreateSource`

Raised at the end of `CreateSource(...)` (after autocomplete/order match/optional attachment download/optional auto-validate).

Signature:

```
[IntegrationEvent(false, false)]
local procedure OnAfterCreateSource(DocHeader: Record "DXP SQZ Document Header"; RawJson: JsonObject)
```

Typical use cases:

- start downstream automations (workflows, notifications)
- additional data validation or custom status logic

Plausibility check events

These events relate to the `IsSourceDataPlausible(...)` method and are relevant if you extend/replace plausibility checks.

Signatures (excerpt):

```
[IntegrationEvent(false, false)]
local procedure OnBeforeDoPlausibilityChecks(DocHeader: Record "DXP SQZ Document Header"; var TempPlausibilityCheckEntry: Record "DXP Plausibility Check Entry" temporary; var IsHandled: Boolean)

[IntegrationEvent(false, false)]
local procedure OnBeforeDoHeaderPlausibilityChecks(DocHeader: Record "DXP SQZ Document Header"; var PlausibilityCheck: Codeunit "DXP Plausiblity Check Mgt.")
```

```
[IntegrationEvent(false, false)]
local procedure OnBeforeDoLinePlausibilityChecks(DocHeader: Record "DXP SQZ Document Header";
DocLine: Record "DXP SQZ Document Line"; var PlausibilityCheck: Codeunit "DXP Plausiblity
Check Mgt.")

[IntegrationEvent(false, false)]
local procedure OnAfterDoPlausibilityChecks(DocHeader: Record "DXP SQZ Document Header"; var
TempPlausibilityCheckEntry: Record "DXP Plausibility Check Entry" temporary)
```

JSON creation (Processed JSON from the SQZ document)

The implementation contains events before line JSON is added to the header structure:

`OnBeforeAddLineToDocumentJObj`

```
[IntegrationEvent(false, false)]
local procedure OnBeforeAddLineToDocumentJObj(var DocumentJObj: JsonObject; DocHeader: Record
"DXP SQZ Document Header")
```

`OnBeforeAddLineJObjToLineJArray`

```

[IntegrationEvent(false, false)]
local procedure OnBeforeAddLineJObjToLineJArray(var LineJObj: JsonObject; DocLine: Record "DXP
SQZ Document Line")
```

Use cases:

- write additional fields into the header JSON
- add additional fields per line (custom tokens)

Events in `DXP SQZ P. Order Conf. Impl.` (document class: Order Confirmation)

This codeunit has **no** dedicated `OnAfterCreateSource` events in `CreateSource(...)`. Extensions here typically happen via:

- `DXP SQZ Document Mgt.` (header/line creation, order match)
- the JSON creation (`CreateProcessedJsonFromSource`)
- plausibility check events

JSON creation

```
[IntegrationEvent(false, false)]
local procedure OnBeforeAddLineToDocumentJObj(var DocumentJObj: JsonObject; DocHeader: Record
"DXP SQZ Document Header")

[IntegrationEvent(false, false)]
local procedure OnBeforeAddLineJObjToLineJArray(var LineJObj: JsonObject; DocLine: Record "DXP
SQZ Document Line")
```

Plausibility check events

Analogous to the Invoice/Credit Memo implementation:

- `OnBeforeDoPlausibilityChecks`
- `OnBeforeDoHeaderPlausibilityChecks`
- `OnBeforeDoLinePlausibilityChecks`
- `OnAfterDoPlausibilityChecks`

Events in `DXP SQZ Document Mgt.` (common to all document classes)

These events are especially relevant because they are raised **within** the header/line creation and order match pipeline.

`OnBeforeModifySQUEEZEDocumentHeader`

Raised in `CreateSQUEEZEDocHeader(...)`, after values have been assigned via field mapping and **before** the header is finally modified.

Signature:

```
[IntegrationEvent(false, false)]
local procedure OnBeforeModifySQUEEZEDocumentHeader(var DocHeader: Record "DXP SQZ Document Header"; RawJson: JsonObject)
```

Use cases:

- adjust header fields before the standard post-processing runs
- take additional values from `RawJson` into the header

`OnBeforeTransferRecognizedValuesOnAfterAssignRecognizedValues`

`OnAfterTransferRecognizedValuesOnBeforeInsertDocLine`

Raised in the standard line post-processing (for standard document classes) around `TransferRecognizedValues(DocLine)`.

Signatures:

```
[IntegrationEvent(false, false)]
local procedure OnBeforeTransferRecognizedValuesOnAfterAssignRecognizedValues(var DocLine: Record "DXP SQZ Document Line"; RowJTok: JsonToken)

[IntegrationEvent(false, false)]
local procedure OnAfterTransferRecognizedValuesOnBeforeInsertDocLine(var DocLine: Record "DXP SQZ Document Line")
```

Use cases:

- correct/normalize recognized values per line
- prepare additional derivations (for example account assignment, dimensions)

`OnBeforePerformAutomaticOrdermatch`

Raised in `PerformAutomaticOrdermatch(...)`, after the order number list (`OrderNoList`) has been filled and **before** the automatic order match runs.

Signature:

```
[IntegrationEvent(false, false)]
local procedure OnBeforePerformAutomaticOrdermatch(DocHeader: Record "DXP SQZ Document Header"; var OrderNoList: List of [Code[20]]; var IsHandled: Boolean)
```

Use cases:

- take over the automatic order match logic completely (`IsHandled := true`)
- adjust the OrderNoList (for example filter/extend)

Events in `DXP SQZ API Mgt.` (close to creation/attachments)

These events sit "around" the API communication (for example attachments).

`OnAfterInitMapping`

Raised in `DownloadFieldMapping(...)`, **after** the document class has initialized the default mapping via the `DXP ISource Document` interface (`ISourceDocument.InitFieldMapping(...)`) and **before** the field list from SQUEEZE is synchronized into the BC field mapping structures.

Signature:

```
[IntegrationEvent(false, false)]
local procedure OnAfterInitMapping(var HeaderMapping: Dictionary of [Text, Integer]; var LineMapping: Dictionary of [Text, Integer]; DocClass: Enum "DXP Document Class")
```

Parameters:

- `HeaderMapping` / `LineMapping`: dictionaries with the default mapping **SQZ field name → BC field no.** (modifiable via `var`)
- `DocClass`: the document class for which the mapping is loaded

Typical use cases:

- add additional (customer-specific) field names as default mapping
- correct the default mapping (for example when SQZ field names have changed)
- mapping validation/logging per document class

`OnAfterSaveAttachment`

Raised in `SaveAttachments(...)` **per attachment**, after the attachment has been created via `DXP Document Attachment Mgt.` and – if the attachment is marked as origin file – after the `Modify(true)` of the record.

Signature:

```
[IntegrationEvent(false, false)]
local procedure OnAfterSaveAttachment(DocumentAttachment: Record "DXP Document Attachment")
```

> `DocumentAttachment` is **not** passed as `var`. If you want to change fields on the record persistently, you must reload the record in the subscriber and explicitly call `Modify(...)`.

Typical use cases:

- index/archive the attachment externally
- derive metadata (for example classification by MIME type/file extension)
- trigger follow-up processes (for example OCR/preview generation in a third-party system)

`OnAfterSetOriginFileNameOnBeforeModifyDocumentAttachment`

Raised in `SaveAttachments(...)`, **when** the attachment is marked as origin file according to the payload and after `Is Source File` and `File Name` (origin name) have been set – but **before** `DocumentAttachment.Modify(true)` is called.

Signature:

```
[IntegrationEvent(false, false)]
local procedure OnAfterSetOriginFileNameOnBeforeModifyDocumentAttachment(var
DocumentAttachment: Record "DXP Document Attachment")
```

Typical use cases:

- normalize the origin file name (for example remove forbidden characters, enforce a naming scheme)
- set additional fields on the attachment before it is saved

Minimal examples: EventSubscriber for `DXP SQZ API Mgt.`

The following examples show typical, **small** extensions. Adjust the object IDs/names to your namespace.

```
codeunit 50101 "My SQZ API Mgt. Subs"
{
    [EventSubscriber(ObjectType::Codeunit, Codeunit::"DXP SQZ API Mgt.", 'OnAfterInitMapping',
'', false, false)]
    local procedure OnAfterInitMapping(var HeaderMapping: Dictionary of [Text, Integer]; var
LineMapping: Dictionary of [Text, Integer]; DocClass: Enum "DXP Document Class")
    var
        SqzDocHeader: Record "DXP SQZ Document Header";
        SqzDocLine: Record "DXP SQZ Document Line";
    begin
        // Example: add a new SQZ field as default mapping.
        // Important: The integer value is the BC field no. of the target field (depends on
your mapping concept).
        if not HeaderMapping.ContainsKey('my_custom_header_token') then
            HeaderMapping.Add('my_custom_header_token', SqzDocHeader.FieldNo("Vendor No."));

        if not LineMapping.ContainsKey('my_custom_line_token') then
            LineMapping.Add('my_custom_line_token', SqzDocLine.FieldNo(Description));
    end;

    [EventSubscriber(ObjectType::Codeunit, Codeunit::"DXP SQZ API Mgt.",
'OnAfterSetOriginFileNameOnBeforeModifyDocumentAttachment', '', false, false)]
    local procedure OnAfterSetOriginFileNameOnBeforeModifyDocumentAttachment(var
DocumentAttachment: Record "DXP Document Attachment")
    begin
        // Example: normalize the origin file name (remove whitespace).
        DocumentAttachment."File Name" := DelChr(DocumentAttachment."File Name", '=', ' ');
    end;

    [EventSubscriber(ObjectType::Codeunit, Codeunit::"DXP SQZ API Mgt.",
'OnAfterSaveAttachment', '', false, false)]
    local procedure OnAfterSaveAttachment(DocumentAttachment: Record "DXP Document
Attachment")
    begin
        // Example: kick off post-processing.
        // If you want to change the attachment persistently, you can also modify the
parameter record directly.
        // DocumentAttachment.Validate("Your Field", ...);
```

```
        // DocumentAttachment.Modify(true);
    end;
}
```

Revision #1
Created 2026-07-22 07:21:43 UTC by Bernd Feddersen
Updated 2026-08-04 09:54:24 UTC by Bernd Feddersen