Events nach der Erstellung eines Core-Belegs (DXP Document)

Zielgruppe

Diese Dokumentation richtet sich an Drittanbieter-Entwickler, die den Lebenszyklus des Core-Belegs **`DXP Document`** erweitern möchten – insbesondere an den Punkten, an denen:

- Status / „Nächster Prozessschritt“ bewertet wird,
- `JSON Raw` / `JSON Processed` aktualisiert wird,
- Standardbelege erzeugt werden (z. B. Einkaufsbelege),
- Attachments übertragen werden.

Einordnung: Core-Beleg-Erstellung vs. Weiterverarbeitung

1) Core-Beleg wird angelegt (`AddDocument`)

Die Anlage eines `DXP Document` erfolgt in **`DXP Document Mgt.`** über `AddDocument(...)`:

- Status wird auf **Imported** gesetzt.
- `JSON Raw` wird (falls vorhanden) in ein BLOB-Feld geschrieben.

2) Danach: Transfer/Verknüpfung und Verarbeitung

Typische nächste Schritte (abhängig vom App-Flow):

- Verknüpfung mit einem Quellbeleg (z. B. SQUEEZE Header) über `UpdateDocumentAfterTransfer(...)`
- Erzeugung eines Standardbelegs und Schreiben des `JSON Processed` über `UpdateDocument(...)`

Die meisten Erweiterungspunkte liegen daher **nicht direkt beim Insert**, sondern im Update-/Processing-Flow.

Event-Matrix (Kurzüberblick)

Bereich	Event	Zeitpunkt	Steuerung	Typische Use-Cases
`DXP Document Mgt.`	`OnAfterSetLinkedToRecIdOnBeforeModify`	in `UpdateDocumentAfterTransfer(...)` vor `Modify(true)`	–	Zusatzinfos setzen, sobald `Linked-to Record Id` steht
`DXP Document Mgt.`	`OnBeforeGetDocumentOnBeforeCheckStatusAndNextProcessStep`	früh in `UpdateDocument(...)` vor `Document.Get(DocNo)`	`IsHandled`	Status/NextStep überschreiben, Processing übernehmen
`DXP Document Mgt.`	`OnAfterGetDocumentOnBeforeCheckStatusAndNextProcessStep`	in `UpdateDocument(...)` nach `Get(...)` vor Standardbeleg-Erzeugung	`Handled`	abhängig vom Dokument entscheiden, JSON/Status anpassen
`DXP Document Mgt.`	`OnAfterUpdateRawJson`	nach Schreiben von `JSON Raw` in `UpdateRawJsonJson(...)`	–	Raw JSON normalisieren/zusätzliche Ableitungen
`DXP Document Mgt.`	`OnAfterWriteProcessedJsonToBlob`	nach Schreiben von `JSON Processed` in `UpdateStatusAndProcessedJson(...)`	–	Processed JSON finalisieren, Tokens ergänzen
`DXP Document Mgt.`	`OnBeforeTransferCoreAttachmentsToStandardDocument`	vor Attachment-Transfer	`IsHandled`	Transfer ersetzen/selektiv steuern
`DXP Document Transfer Mgt.`	`OnBeforeExitProcessedJsonDocHeader`	beim Erzeugen von Header-JSON (vor `exit(JObject)`)	–	Standard-Tokens ergänzen/normalisieren
`DXP Document Transfer Mgt.`	`OnBeforeExitProcessedJsonDocLine`	beim Erzeugen von Line-JSON (vor `exit(JObject)`)	–	Line-Tokens ergänzen/normalisieren
`DXP Document Transfer Mgt.`	`OnAfterCreatePurchaseHeader`	nach Anlage/Validierung des Purchase Headers	–	Header-Felder ergänzen, Post-Processing
`DXP Document Transfer Mgt.`	`OnBeforeInsertPurchaseLine`	vor Insert einer Purchase Line	–	Purchase Line anpassen (z. B. Kontierung, Typ/No)

Bereich	Event	Zeitpunkt	Steuerung	Typische Use-Cases
`DXP Document Transfer Mgt.`	`OnAfterInsertPurchaseLineOnBeforeAddDimensions`	nach Insert, vor Dimensionen	–	Dimensions/Additional-Info vorbereiten
`DXP Document Transfer Mgt.`	`OnAfterOnBeforeAddLineDimensions`	bei Receipt-Lines: vor Dimensionen des zu aktualisierenden Lines	–	Update-Fälle (Receipt) anpassen
`DXP Document Transfer Mgt.`	`OnAfterProcessAdditionalInformation`	nach Transfer zusätzlicher Informationen	–	Add-on-Felder pro Ziel-RecordRef ergänzen
`DXP Document Transfer Mgt.`	`OnAfterCreatePurchaseDoc`	nach kompletter Erstellung des Purchase Dokuments	–	finaler Hook (Logging, Folgeaktionen)

Events in `DXP Document Mgt.` (Core-Lebenszyklus)

`OnAfterSetLinkedToRecIdOnBeforeModify`

Wird in `UpdateDocumentAfterTransfer(...)` ausgelöst – nachdem Status/`Linked-to Record Id` gesetzt wurden, aber **bevor** das `DXP Document` gespeichert wird.

Signatur:

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

Use-Cases:

- Zusatzinformationen am Core-Beleg setzen, sobald er auf einen Quellbeleg verlinkt ist
- eigene Logging-/Audit-Informationen ergänzen

`OnBeforeGetDocumentOnBeforeCheckStatusAndNextProcessStep`

Wird in `UpdateDocument(...)` als **frühester Hook** aufgerufen – noch bevor das `DXP Document` per `Get(DocNo)` geladen wird.

Signatur:

```
[IntegrationEvent(false, false)]
local procedure OnBeforeGetDocumentOnBeforeCheckStatusAndNextProcessStep(
    DocNo: Code[20];
    var Status: Enum "DXP Document Status";
    var NextStep: Enum "DXP Next Process Step";
    JObject: JsonObject;
    var IsHandled: Boolean)
```

Use-Cases:

- Status/NextStep vorab überschreiben (z. B. „Individuelle Verarbeitung“)
- komplette Verarbeitung übernehmen (`IsHandled := true`)

`OnAfterGetDocumentOnBeforeCheckStatusAndNextProcessStep`

Wird in `UpdateDocument(...)` nach `Document.Get(DocNo)` ausgelöst, bevor Standardbelege erzeugt werden.

Signatur:

```
[IntegrationEvent(false, false)]
local procedure OnAfterGetDocumentOnBeforeCheckStatusAndNextProcessStep(
    var Document: Record "DXP Document";
    NewStatus: Enum "DXP Document Status";
    NextStep: Enum "DXP Next process step";
    var ProcessedJSONObj: JsonObject;
    var Handled: Boolean)
```

Use-Cases:

- abhängig vom aktuellen Core-Dokument entscheiden, ob Processing übernommen werden soll
- Anpassungen am `ProcessedJSONObj` vor der weiteren Verarbeitung

`OnAfterUpdateRawJson`

Wird in `UpdateRawJsonJson(...)` nach dem Schreiben von `JSON Raw` ausgelöst.

Signatur:

```
[IntegrationEvent(false, false)]
local procedure OnAfterUpdateRawJson(var Document: Record "DXP Document"; var RawJSONObj: JsonObject)
```

Use-Cases:

- Validierungs-/Normalisierungsschritte auf `RawJSONObj`
- zusätzliche Datenextraktion direkt nach Raw-Update

`OnAfterWriteProcessedJsonToBlob`

Wird in `UpdateStatusAndProcessedJson(...)` ausgelöst, nachdem `JSON Processed` in das BLOB geschrieben wurde.

Signatur:

```
[IntegrationEvent(false, false)]
local procedure OnAfterWriteProcessedJsonToBlob(var Document: Record "DXP Document"; var ProcessedJSONObj: JsonObject)
```

Use-Cases:

- zusätzliche Tokens ergänzen
- Datenstruktur finalisieren, bevor nachgelagerte Prozesse darauf zugreifen

`OnBeforeTransferCoreAttachmentsToStandardDocument`

Wird in `TransferCoreAttachmentsToStandardDocument(...)` vor dem Transfer ausgelöst. Über `IsHandled` kann die Standardlogik ersetzt werden.

Signatur:

```
[IntegrationEvent(false, false)]
local procedure OnBeforeTransferCoreAttachmentsToStandardDocument(Document: Record "DXP Document"; var IsHandled: Boolean)
```

Use-Cases:

- Attachment-Transfer komplett übernehmen (z. B. andere Ablage/Benennung)
- Transfer selektiv einschränken

Events in `DXP Document Transfer Mgt.` (Standardbelege aus `JSON Processed`)

Diese Codeunit ist relevant, sobald aus `JSON Processed` Standardbelege (z. B. Einkaufsbelege) erzeugt werden.

`OnBeforeExitProcessedJsonDocHeader`

Wird in `CreateDocHeaderJson(...)` unmittelbar vor `exit(JObject)` ausgelöst.

Signatur:

```
[IntegrationEvent(false, false)]
local procedure OnBeforeExitProcessedJsonDocHeader(var JObject: JsonObject)
```

Use-Cases:

- zusätzliche Standard-Tokens ergänzen
- Token-Werte normalisieren (z. B. Typ-/Textdarstellung)

`OnBeforeExitProcessedJsonDocLine`

Wird in `CreateDocLineJson(...)` unmittelbar vor `exit(JObject)` ausgelöst.

Signatur:

```
[IntegrationEvent(false, false)]
local procedure OnBeforeExitProcessedJsonDocLine(var JObject: JsonObject)
```

Use-Cases:

- Line-Tokens ergänzen/normalisieren

`OnAfterCreatePurchaseHeader`

Wird nach Anlage/Validierung des Purchase Headers ausgelöst.

Signatur:

```
[IntegrationEvent(false, false)]
local procedure OnAfterCreatePurchaseHeader(var JObject: JsonObject; PurchaseHeader: Record
"Purchase Header")
```

Use-Cases:

- zusätzliche Felder am `Purchase Header` setzen (z. B. eigene Referenzen)
- Post-Processing/Logging

`OnBeforeInsertPurchaseLine`

Wird unmittelbar vor dem Insert einer Purchase Line ausgelöst.

Signatur:

```
[IntegrationEvent(false, false)]
local procedure OnBeforeInsertPurchaseLine(var JObjectLine: JsonObject; var PurchaseLine:
```

```
Record "Purchase Line")
```

Use-Cases:

- Purchase Line anpassen (z. B. Kontierung, Typ/No)

OnAfterInsertPurchaseLineOnBeforeAddDimensions

Wird nach Insert der Purchase Line, aber vor Dimensions-Validierung ausgelöst.

Signatur:

```
[IntegrationEvent(false, false)]
local procedure OnAfterInsertPurchaseLineOnBeforeAddDimensions(var JObjectLine: JsonObject;
PurchaseLine: Record "Purchase Line")
```

Use-Cases:

- Dimensions/Additional-Info vorbereiten

OnAfterOnBeforeAddLineDimensions

Wird im Receipt-Line-Update-Fall ausgelöst, bevor Dimensionen für die zu aktualisierende Purchase Line gesetzt werden.

Signatur:

```
[IntegrationEvent(false, false)]
local procedure OnAfterOnBeforeAddLineDimensions(var JObjectLine: JsonObject;
PurchaseLineToUpdate: Record "Purchase Line")
```

Use-Cases:

- Update-Fälle (Receipt) anpassen

`OnAfterProcessAdditionalInformation`

Wird nach dem Transfer zusätzlicher Informationen (`ProcessAdditionalInformation`) ausgelöst.

Signatur:

```
[IntegrationEvent(false, false)]
local procedure OnAfterProcessAdditionalInformation(var RecRef: RecordRef; SourceJsonObject:
JsonObject)
```

Use-Cases:

- Add-on Felder für unterschiedliche Zieltabellen (Header/Line) ergänzen

`OnAfterCreatePurchaseDoc`

Wird nach kompletter Erstellung des Purchase Dokuments ausgelöst.

Signatur:

```
[IntegrationEvent(false, false)]
local procedure OnAfterCreatePurchaseDoc(var JObject: JsonObject; PurchaseHeader: Record
"Purchase Header")
```

Use-Cases:

- finaler Hook (Post-Processing, Logging, Folgeaktionen)

Minimalbeispiel: Status/NextStep in Richtung „Individuelle Verarbeitung“ beeinflussen

Beispiel: Vor dem Standard-Processing `Status` auf „Custom Processing“ setzen:

```
codeunit 50101 "My Core Document Hooks"
{
```

```
    [EventSubscriber(ObjectType::Codeunit, Codeunit::"DXP Document Mgt.",
'OnBeforeGetDocumentOnBeforeCheckStatusAndNextProcessStep', '', false, false)]
    local procedure OnBeforeGetDoc(
        DocNo: Code[20];
        var Status: Enum "DXP Document Status";
        var NextStep: Enum "DXP Next Process Step";
        JObject: JsonObject;
        var IsHandled: Boolean)
    begin
        Status := Status::"Custom Processing";
        // NextStep optional anpassen
    end;
}
```

[Best Practice] Nutze `IsHandled := true` nur, wenn du die Standardlogik vollständig ersetzt – sonst kann es zu inkonsistenten Status-/JSON-Zuständen kommen.

Revision #2
Created 2026-02-03 12:19:51 UTC by Bernd Feddersen
Updated 2026-02-03 12:23:58 UTC by Bernd Feddersen