Squeeze-Ursprungsanhang anpassen

Übersicht

Das `OnAfterSetOriginFileNameOnBeforeModifyDocumentAttachment` Integration Event ermöglicht es Drittanbieter-Entwicklern, Dokumentenanhang-Datensätze anzupassen, bevor sie während des SQUEEZE-Anhang-Verarbeitungsworkflows modifiziert werden. Ein Ursprungsanhang ist der relevante Anhang, der zur Erstellung eines Belegs führt (z.B. die Rechnung des Lieferanten).

Event-Deklaration

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

Event-Parameter

Parameter	Typ	Beschreibung
DocumentAttachment	`Record "DXP Document Attachment"` (var)	Der Dokumentenanhang-Datensatz, der modifiziert werden soll. Als Referenz übergeben, wodurch Änderungen möglich sind.

Wann wird dieses Event ausgelöst?

Dieses Event wird während der `SaveAttachments`-Prozedur ausgelöst, wenn:

- Ein Anhang von SQUEEZE verarbeitet wird

- Der Anhang als Ursprungsdatei identifiziert wird (`IsOriginFile = true`)
- Das System den Ursprungsdateinamen und das `DXP Is Origin File` Flag gesetzt hat
- Kurz vor dem `DocumentAttachment.Modify(true)`-Aufruf

Anwendungsfälle

Dieses Integration Event ist nützlich für:

- **Benutzerdefiniertes Füllen von Feldern**: Setzen zusätzlicher benutzerdefinierter Felder im Dokumentenanhang
- **Dateinamen-Transformation**: Anwenden benutzerdefinierter Namenskonventionen oder Formatierung
- **Metadaten-Erweiterung**: Hinzufügen benutzerdefinierter Metadaten oder Tags zu Anhängen
- **Validierungslogik**: Implementierung benutzerdefinierter Validierung vor dem Speichern des Datensatzes
- **Integrationsanforderungen**: Vorbereitung von Daten für externe Systemintegrationen

Implementierungsbeispiel

Dateinamen vor Modifikation ändern

```
[EventSubscriber(ObjectType::Codeunit, Codeunit::"DXP SQZ API Mgt.",
'OnAfterSetOriginFileNameOnBeforeModifyDocumentAttachment', '', false, false)]
local procedure OnAfterSetOriginFileNameOnBeforeModifyDocumentAttachment(var
DocumentAttachment: Record "DXP Document Attachment")
var
    CoreDocument: Record "DXP Document";
    SQZDocumentHeader: Record "DXP SQZ Document Header";
    Vendor: Record Vendor;
    NewFileName: Text[250];
    VendorPrefix: Text[50];
begin
    // Get the core document using the document attachment's Document No.
    if not CoreDocument.Get(DocumentAttachment."Document No.") then
        exit;

    // Check if the core document has a linked SQUEEZE document
```

```
    if IsNullGuid(CoreDocument."Linked-to Record Id") then
        exit;

    // Get the SQZ Document Header using the SystemId from Linked-to Record Id
    if not GetSQZDocumentHeaderFromRecordId(CoreDocument."Linked-to Record Id",
SQZDocumentHeader) then
        exit;

    // Check if Buy-from Vendor No. is populated
    if SQZDocumentHeader."Buy-from Vendor No." = '' then
        exit;

    // Get the vendor record
    if not Vendor.Get(SQZDocumentHeader."Buy-from Vendor No.") then
        exit;

    // Create vendor prefix from Search Name (fallback to Name if Search Name is empty)
    if Vendor."Search Name" <> '' then
        VendorPrefix := Vendor."Search Name"
    else
        VendorPrefix := Vendor.Name;

    // Clean the vendor prefix (remove invalid filename characters and limit length)
    VendorPrefix := CleanFilenameText(VendorPrefix, 30);

    // Check if vendor prefix already exists in filename to avoid duplicates
    if DocumentAttachment."File Name".StartsWith(VendorPrefix + '_') then
        exit;

    // Create new filename with vendor prefix
    NewFileName := VendorPrefix + '_' + DocumentAttachment."File Name";

    // Update the attachment filename
    DocumentAttachment."File Name" := NewFileName;
end;

local procedure GetSQZDocumentHeaderFromRecordId(LinkedRecordId: Guid; var SQZDocumentHeader:
Record "DXP SQZ Document Header"): Boolean
var
```

```
    RecRef: RecordRef;
    SystemIdFieldRef: FieldRef;
begin
    // Method 1: Try to get the record directly if LinkedRecordId is actually a SystemId
    if SQZDocumentHeader.GetBySystemId(LinkedRecordId) then
        exit(true);

    // Method 2: If that fails, we need to find the record another way
    // This assumes the Linked-to Record Id might be stored differently
    SQZDocumentHeader.Reset();
    SQZDocumentHeader.SetRange(SystemId, LinkedRecordId);
    exit(SQZDocumentHeader.FindFirst());
end;

local procedure CleanFilenameText(InputText: Text; MaxLength: Integer): Text
var
    CleanText: Text;
begin
    // Remove invalid filename characters
    CleanText := DelChr(InputText, '=', '<>|"/\:*?');

    // Replace spaces with underscores for better filename compatibility
    CleanText := CleanText.Replace(' ', '_');

    // Limit length
    CleanText := CopyStr(CleanText, 1, MaxLength);

    exit(CleanText);
end;
```

Wichtige Überlegungen

Datenintegrität

- Der `DocumentAttachment`-Datensatz wird als Referenz übergeben, daher werden alle Änderungen gespeichert
- Berücksichtigen Sie Feldlängenbegrenzungen beim Ändern von Textfeldern

Performance

- Halten Sie die Verarbeitung schlank, da dieses Event für jeden Ursprungsdatei-Anhang aufgerufen wird
- Erwägen Sie das Zwischenspeichern häufig verwendeter Daten

Fehlerbehandlung

- Implementieren Sie ordnungsgemäße Fehlerbehandlung, um zu verhindern, dass der Anhang-Speichervorgang fehlschlägt
- Verwenden Sie try-Funktionen für riskante Operationen

Verwandte Events

- `OnAfterSaveAttachment`: Wird ausgelöst, nachdem jeder Anhang vollständig gespeichert wurde.
- Erwägen Sie die Verwendung dieses alternativen Events, wenn Sie Aktionen nach dem Speichern des Datensatzes durchführen müssen.

Fehlerbehebung

Wenn Ihr Event Subscriber nicht ausgelöst wird:

- Testen Sie speziell mit Ursprungsdateien (Nicht-Ursprungsdateien lösen dieses Event nicht aus)

Revision #3
Created 2025-05-28 09:42:52 UTC by Bernd Feddersen
Updated 2026-01-29 09:10:04 UTC by Bernd Feddersen