Embedding the SQUEEZE Viewer part in your own pages WIP

Diese Anleitung beschreibt, wie Sie den **SQUEEZE Viewer** in *beliebige* Seiten einbetten (eigene Pages oder Page Extensions) – über den wiederverwendbaren Viewer-Part:

- Page: **"DXP SQZ Viewer Part"** (PageType = `CardPart`)
- Source table: **"DXP SQZ Document Header"**

Der Viewer-Part hostet die SQUEEZE Control Add-ins und lädt die Preview-URL anhand des aktuellen Datensatzes **DXP SQZ Document Header**.

Referenz-Implementierung, die in dieser App bereits enthalten ist:

- Page **"DXP SQZ Document (Generic)"** (Card, SourceTable **DXP SQZ Document Header**) bettet den Viewer-Part als FactBox ein und verdrahtet das Resident-Sharing zur generischen Lines-Subform.
- Page **"DXP SQZ Doc. Generic Subform"** (ListPart, SourceTable **DXP SQZ Document Line**) enthält eine bekannte, funktionierende Minimal-Implementierung von `MarkRow()`.

Die Pages können Sie benutzen und mit ihren Feldern erweitern. Die Page **"DXP SQZ Document (Generic)"** verfügt über einen Viewer in der Factbox und die wichtigsten Actions.

---

1) Voraussetzungen

1. Ihre Extension muss von der DEXPRO SQUEEZE App abhängig sein (damit Page/Control Add-ins als Symbole verfügbar sind).
2. Der Host-Datensatz muss eine Möglichkeit bieten, einen **DXP SQZ Document Header** Datensatz zu referenzieren.
   - Die `SourceTable` des Viewer-Parts ist fest auf **DXP SQZ Document Header**.
   - Deshalb muss Ihre Seite ein Feld haben (oder bereitstellen), das sich in `SubPageLink` verwenden lässt, um den Header zu finden. Alternativ kann der Record in dem Viewer-Part gesetzt werden - Beispiele folgen weiter unten.
3. SQUEEZE muss aktiviert und konfiguriert sein (Setup + API Key). Der Viewer-Part lädt absichtlich **nicht**, wenn das Modul nicht aktiviert ist.

---

2) Empfohlenes Einbettungsmuster

2.1 Als FactBox einbetten (empfohlen)

Für die meisten Szenarien (Card Pages, List Pages) platzieren Sie den Viewer in `area(FactBoxes)`.

Wenn die `SourceTable` Ihrer Seite **DXP SQZ Document Header** ist

```
layout
{
    area(FactBoxes)
    {
        part(SQZViewer; "DXP SQZ Viewer Part")
        {
            ApplicationArea = All;
            Caption = 'Viewer';

            // Link the FactBox part to the current SQZ header
            SubPageLink = "No." = field("No.");
        }
    }
}
```

Wenn die `SourceTable` Ihrer Seite NICHT **DXP SQZ Document Header** ist

Sie können den Viewer-Part trotzdem einbetten, sofern Ihr Datensatz ein Feld enthält, das die SQUEEZE Header-Nummer speichert.

Beispiel: Ihr Datensatz hat ein Feld `"SQZ Document No." : Code[20]`

```
layout
{
    area(FactBoxes)
    {
        part(SQZViewer; "DXP SQZ Viewer Part")
        {
            ApplicationArea = All;
            Caption = 'Viewer';
```

```
            // Link the viewer part's "No." to your field
            SubPageLink = "No." = field("SQZ Document No.");
        }
    }
}
```

Wichtig:

- Setze am Part nicht `Enabled = false`. Wenn eine FactBox-Part deaktiviert ist, kann das verhindern, dass das Control Add-in „ready“ wird – dadurch bleibt der Viewer ggf. dauerhaft leer.

2.2 Viewer dynamisch binden (ohne `SubPageLink`)

Wenn Ihre Host-Seite keinen einfachen `SubPageLink` anbieten kann (z. B. weil es sich um eine Zielbeleg-Seite wie **Purchase Invoice** handelt und dort die SQUEEZE Belegnummer nicht gespeichert ist), können Sie den Viewer-Part trotzdem einbetten und im Code binden.

Konzept:

- Viewer-Part als FactBox hinzufügen – **ohne** `SubPageLink`.
- In `OnAfterGetCurrRecord()` den relevanten `Record "DXP SQZ Document Header"` auflösen.
- Den Datensatz per `CurrPage.<PartName>.Page.SetSqzRecord(SqzHeader);` in den Part pushen.

Warum `SetSqzRecord`?

- Im DEXPRO SQUEEZE Viewer-Part läuft das Laden der URL in `OnAfterGetCurrRecord()`.
- Beim Navigieren innerhalb einer Seite (z. B. Wechsel zwischen Purchase Invoices) kann `SetRecord(...)` allein dazu führen, dass die FactBox nicht sofort neu lädt.
- `SetSqzRecord(...)` ist dafür gedacht, den Header sauber neu zu binden und einen Refresh auszulösen, sodass die korrekte Preview ohne Schließen/Öffnen der Seite geladen wird.

Beispiel-Gerüst:

```
layout
{
    area(FactBoxes)
    {
        part(SQZViewer; "DXP SQZ Viewer Part")
        {
            ApplicationArea = All;
            Caption = 'Viewer';
```

```
        }
    }
}

trigger OnAfterGetCurrRecord()
var
    SqzHeader: Record "DXP SQZ Document Header";
begin
    if TryResolveSqzHeaderForCurrentRecord(SqzHeader) then
        CurrPage.SQZViewer.Page.SetSqzRecord(SqzHeader);
end;
```

Dieses Pattern ist besonders für Third-party Pages hilfreich, weil es vermeidet, „SQUEEZE Document No.“ als Tabellenfeld persistieren zu müssen.

---

3) Viewer?Modus (Belegübersicht / Positionsfokus / Abgedockt)

Wenn der Viewer als FactBox eingebettet ist, sind Felder typischerweise nicht editierbar. Deshalb bietet der Viewer-Part **Actions**, um den Viewer-Modus zu ändern.

Was beim Wechsel des Modus passiert:

- **Belegübersicht**: Zeigt eine Übersichtliche Darstellung des Viewers an.
- **Positionsfokus**: Zeigt eine schmalere Darstellung des Viewers an.
- **Abgedockt**: Öffnet den Viewer in einem externen Fenster

Für Third-party Entwickler:

- Sie müssen keine eigene Mode-Logik implementieren.
- Nutzen Sie die Actions des Viewer-Parts (erscheinen im FactBox Action-Menü).

---

4) Optional, aber wichtig: Zeilen? Highlighting aktivieren (MarkRow)

Wenn Sie eine Lines-Subform haben und die Benutzererfahrung „Zeile klicken → Viewer markiert Koordinaten“ wünschen, muss Ihre Lines-Page Messages über **dieselbe**

Viewer-Resident-Instanz posten, die der Viewer-Part initialisiert.

Warum das wichtig ist:

- Die Viewer-UI ist eine Control Add-in Instanz, die in einer Codeunit `DXP SQUEEZE Viewer Resident` gespeichert wird.
- Wenn Ihre Lines-Page eine *eigene* Variable `DXP SQUEEZE Viewer Resident` verwendet (die nicht vom Viewer-Part initialisiert wurde), hat `ApiMgt.PostMessage(...)` keine Control-Add-in Instanz zum Ansprechen – es wird nichts markiert.

4.1 Pattern (Host?Page verdrahtet Viewer? Resident in Lines)

Annahmen:

- Ihre Header-Page hat:
  - `part(SQZViewer; "DXP SQZ Viewer Part")`
  - `part(DocumentLines; <Ihre Lines ListPart>)`
- Ihre Lines-ListPart stellt eine Procedure bereit:
  - `SetViewerResident(var NewResident: Codeunit "DXP SQUEEZE Viewer Resident")`

Das ist ein expliziter Vertrag/Annahme für interaktives Line-Highlighting.

Ergänzen Sie das auf Ihrer Header-Page:

```
trigger OnAfterGetCurrRecord()
var
    SharedResident: Codeunit "DXP SQUEEZE Viewer Resident";
begin
    CurrPage.SQZViewer.Page.GetViewerResident(SharedResident);
    CurrPage.DocumentLines.Page.SetViewerResident(SharedResident);
end;
```

Hinweise:

- Das gehört in `OnAfterGetCurrRecord()` der Host-Page (dort ist der aktuelle Header stabil, und die Parts existieren).
- Wenn Ihre Host-Page andere Part-Namen verwendet, passen Sie `CurrPage.<PartName>.Page...` entsprechend an.

4.2 Was die Lines?Subform tun muss

Ihre Lines-Page sollte `MarkRow()` in `OnAfterGetCurrRecord()` aufrufen und `MarkRow()` ähnlich zur Standard-Seite implementieren:

- Wenn der externe Viewer aktiv ist (Abgedockt), nur die State-Tabelle aktualisieren.
- Ansonsten `ApiMgt.PostMessage(ApiMgt.MarkFieldByCoordinatesObj(Rec.RecordId), Resident)` aufrufen.

Das ist die Mindestanforderung für interaktives Highlighting.

4.3 Referenz?Implementierung: minimale Lines? Subform mit `MarkRow`

Das ist ein bekannt funktionierendes Minimal-Pattern (basierend auf der Standard SQUEEZE Lines-Subform). Third-party Entwickler können es kopieren und Felder/Repeater-Spalten nach Bedarf anpassen:

```
page 70954672 "DXP SQZ Doc. Generic Subform"
{
    Caption = 'Lines';
    PageType = ListPart;
    AutoSplitKey = true;
    DelayedInsert = true;
    SourceTable = "DXP SQZ Document Line";
    RefreshOnActivate = true;

    layout
    {
        area(content)
        {
            repeater(General)
            {
                field("Line No."; Rec."Line No.")
                {
                    ApplicationArea = All;
                    Visible = false;
                }
                // Add your visible SQZ fields here...
            }
        }
    }
```

```
    trigger OnAfterGetCurrRecord()
    begin
        MarkRow();
    end;

    var
        ApiMgt: Codeunit "DXP SQZ API Mgt.";
        Resident: Codeunit "DXP SQUEEZE Viewer Resident";
        MarkedLineNo: Integer;

    procedure SetViewerResident(var NewResident: Codeunit "DXP SQUEEZE Viewer Resident")
    begin
        Resident := NewResident;
        MarkedLineNo := 0;
    end;

    local procedure MarkRow()
    var
        DocHeader: Record "DXP SQZ Document Header";
        LineCoordMgt: Codeunit "DXP SQZ Coordinates Mgt.";
        ViewerState: Codeunit "DXP Viewer State";
    begin
        if MarkedLineNo = Rec."Line No." then
            exit;

        if not LineCoordMgt.CoordinateExists(Rec.RecordId) then
            exit;

        // Detached viewer: update the state table only
        if IsExternalViewerActive() then begin
            if DocHeader.Get(Rec."Document No.") then
                ViewerState.UpdateViewerStateWithCoordinates(DocHeader."API Document ID",
Rec.RecordId);
        end else
            // Internal viewer: post message to the control add-in instance via the shared
resident
            ApiMgt.PostMessage(ApiMgt.MarkFieldByCoordinatesObj(Rec.RecordId), Resident);

        MarkedLineNo := Rec."Line No.";
    end;
```

```
    local procedure IsExternalViewerActive(): Boolean
    var
        ViewerState: Codeunit "DXP Viewer State";
    begin
        exit(ViewerState.GetViewerMode() = "DXP SQZ Viewer Mode"::Detached);
    end;
}
```

---

5) Fehlersuche (Viewer bleibt leer)

5.1 Der Viewer lädt nie eine Preview

Prüfe zuerst:

- Ist das SQUEEZE Modul aktiviert und das Setup vollständig?
- Hat der verknüpfte **DXP SQZ Document Header** Datensatz eine gültige `"API Document ID"`?
- Ist der Viewer-Modus aktuell **Abgedockt** und ein externer Viewer offen?
  - Der interne Viewer lädt nicht, solange der externe Viewer geöffnet ist.

5.2 FactBox ist sichtbar, aber Control Add-in initialisiert nicht

- Deaktiviere den Viewer-FactBox-Part nicht über `Enabled = false`.
- Stelle sicher, dass die FactBox tatsächlich angezeigt wird (FactBoxes können vom Benutzer eingeklappt/ausgeblendet werden).

5.3 MarkRow hebt nichts hervor

- Stelle sicher, dass die Lines-Page dieselbe Viewer-Resident-Instanz wie der Viewer-Part nutzt (siehe Abschnitt 4).
- Stelle sicher, dass Koordinaten für den Zeilen-Datensatz existieren (`DXP SQZ Coordinates Mgt.` muss Koordinaten zu `Rec.RecordId` finden).

---

6) Minimales Beispiel: Viewer in eine eigene Page Extension einbauen

```
pageextension 50100 "MY SQZ Doc Ext." extends "DXP SQZ Document (Generic)"
{
    layout
    {
        addfirst(FactBoxes)
        {
            part(MySQZViewer; "DXP SQZ Viewer Part")
            {
                ApplicationArea = All;
                Caption = 'Viewer';
                SubPageLink = "No." = field("No.");
            }
        }
    }
}
```

---

7) Hinweise / Einschränkungen

- Der Viewer-State ist session-getrieben (via `DXP Viewer State`). Wenn mehrere Seiten mit Viewer gleichzeitig offen sind, können Modus/External-Viewer-State alle beeinflussen.
- Der Viewer-Part ist für Datensätze mit SQUEEZE-Bezug gedacht; er versucht nicht, automatisch „irgendeinen“ Header zu suchen.

---

8) Beispiel: Viewer auf einem Zielbeleg anzeigen (Purchase Invoice)

Dieser Abschnitt adressiert ein häufiges Third-party Szenario:

Annahmen:

- Der Zielbeleg (z. B. **Purchase Invoice**) wurde aus einem Core-Beleg erstellt.
- Der Core-Beleg ist weiterhin in der Tabelle **"DXP Document"** verfügbar.

- Der Core-Beleg speichert eine Rückverknüpfung zum Ziel-Datensatz im Feld **"Linked-to Record Id"**.
- Der SQUEEZE Quell-Header existiert noch und kann aus dem Core-Beleg aufgelöst werden (typischerweise über **Core Document No.**).

8.1 Core?Beleg aus dem Ziel?Datensatz auflösen

```
local procedure TryGetCoreDocumentForTarget(TargetRecId: RecordId; var CoreDoc: Record "DXP Document"): Boolean
begin
    CoreDoc.Reset();
    CoreDoc.SetRange("Linked-to Record Id", TargetRecId);
    exit(CoreDoc.FindFirst());
end;
```

8.2 SQUEEZE Header aus dem Core?Beleg auflösen

```
local procedure TryGetSqzHeaderForCore(CoreDocNo: Code[20]; var SqzHeader: Record "DXP SQZ Document Header"): Boolean
begin
    SqzHeader.Reset();
    SqzHeader.SetRange("Core Document No.", CoreDocNo);
    exit(SqzHeader.FindFirst());
end;
```

8.3 Viewer einbetten (dynamisches Binden)

Beispiel: **Purchase Invoice** erweitern (konzeptionell; Basis-Page-Name ggf. an Ihre Umgebung anpassen).

Diese Implementierung verwendet:

- Einen FactBox-Part
- Ein `ShowViewer`-Flag, um den Part auszublenden, wenn kein SQZ Header aufgelöst werden kann.
- `SetSqzRecord(...)`, damit die Preview beim Wechsel des Belegs sofort korrekt neu lädt.

- Ein `SingleInstance` Context-Codeunit, um Viewer-Resident und SQZ-Kontext mit der **Purch. Invoice Subform** zu teilen (FactBoxes und Subforms sind separate Pages).

```
pageextension 50100 "MY Purch. Invoice Viewer" extends "Purchase Invoice"
{
    layout
    {
        addfirst(FactBoxes)
        {
            part(DXPSQZViewer; "DXP SQZ Viewer Part")
            {
                ApplicationArea = All;
                Caption = 'SQUEEZE Viewer';
                Visible = ShowViewer;
            }
        }
    }

    var
        ShowViewer: Boolean;

    trigger OnAfterGetCurrRecord()
    var
        CoreDoc: Record "DXP Document";
        SqzHeader: Record "DXP SQZ Document Header";
        SharedResident: Codeunit "DXP SQUEEZE Viewer Resident";
        ViewerCtx: Codeunit "DXP SQZ Purch. Inv. Viewer Ctx";
    begin
        ShowViewer := false;
        if not TryGetCoreDocumentForTarget(Rec.RecordId, CoreDoc) then
            exit;

        if not TryGetSqzHeaderForCore(CoreDoc."No.", SqzHeader) then
            exit;

        ShowViewer := true;

        CurrPage.DXPSQZViewer.Page.SetSqzRecord(SqzHeader);

        CurrPage.DXPSQZViewer.Page.GetViewerResident(SharedResident);
```

```
        ViewerCtx.SetResident(SharedResident);
        ViewerCtx.SetSqzContextForPurchInv(Rec."No.", SqzHeader."No.", SqzHeader."API Document
ID");
    end;

    local procedure TryGetCoreDocumentForTarget(TargetRecId: RecordId; var CoreDoc: Record
"DXP Document"): Boolean
    begin
        CoreDoc.Reset();
        CoreDoc.SetRange("Linked-to Record Id", TargetRecId);
        exit(CoreDoc.FindFirst());
    end;

    local procedure TryGetSqzHeaderForCore(CoreDocNo: Code[20]; var SqzHeader: Record "DXP SQZ
Document Header"): Boolean
    begin
        SqzHeader.Reset();
        SqzHeader.SetRange("Core Document No.", CoreDocNo);
        exit(SqzHeader.FindFirst());
    end;
}
```

Zugehöriges Context-Codeunit (speichert Resident + SQZ-Kontext, damit die Subform korrekt highlighten kann):

```
codeunit 50101 "MY SQZ Purch. Inv. Viewer Ctx"
{
    SingleInstance = true;

    procedure SetResident(var NewResident: Codeunit "DXP SQUEEZE Viewer Resident")
    begin
        Resident := NewResident;
        HasResident := true;
    end;

    procedure TryGetResident(var OutResident: Codeunit "DXP SQUEEZE Viewer Resident"): Boolean
    begin
        if not HasResident then
            exit(false);
```

```
        OutResident := Resident;
        exit(true);
    end;

    procedure SetSqzContextForPurchInv(PurchInvNo: Code[20]; SqzDocNo: Code[20];
ApiDocumentId: Text[50])
    begin
        PurchInvToSqzDocNo.Set(PurchInvNo, SqzDocNo);
        PurchInvToApiDocumentId.Set(PurchInvNo, ApiDocumentId);
    end;

    procedure TryGetSqzContextForPurchInv(PurchInvNo: Code[20]; var SqzDocNo: Code[20]; var
ApiDocumentId: Text[50]): Boolean
    begin
        if not PurchInvToSqzDocNo.Get(PurchInvNo, SqzDocNo) then
            exit(false);

        PurchInvToApiDocumentId.Get(PurchInvNo, ApiDocumentId);
        exit(true);
    end;

    var
        Resident: Codeunit "DXP SQUEEZE Viewer Resident";
        HasResident: Boolean;
        PurchInvToSqzDocNo: Dictionary of [Code[20], Code[20]];
        PurchInvToApiDocumentId: Dictionary of [Code[20], Text[50]];
}
```

8.4 Wichtiger Hinweis: SQUEEZE Header? Status nach Verarbeitung

Sie haben ein häufiges Setup beschrieben, in dem der SQUEEZE Header nach dem Erstellen des Zielbelegs noch existiert, aber `Status = Deleted` ist.

Der Viewer-Part kann die Preview trotzdem laden, solange `"API Document ID"` vorhanden ist und SQUEEZE aktiviert ist.

Optionen:

- Wenn Sie die Preview nach Zielbeleg-Erstellung **nicht** anzeigen möchten, bauen Sie eine Guard-Logik auf der Host-Page ein (Viewer-Part dann nicht binden, wenn der SQZ Header „Deleted“ ist).

Welche Option korrekt ist, hängt von Ihren Prozessanforderungen ab.

---

9) Sonderfall: Hervorheben aus der Purchase Invoice Subform (Purchase Line ? SQZ line)

Manchmal mächten Sie diese Benutzererfahrung auf einer Zielbeleg-Seite:

- User navigiert durch **Purchase Lines**.
- Der eingebettete SQUEEZE Viewer hebt den entsprechenden Bereich (Koordinaten) des originalen Belegs, bzw. der Viewer-Abbildung hervor.

In dieser Situation basiert Ihre Lines-Page **nicht** auf `"DXP SQZ Document Line"`, daher können Sie `MarkFieldByCoordinatesObj(Rec.RecordId)` nicht direkt aufrufen.

Stattdessen müssen Sie die passende SQUEEZE Line auflösen und danach mit der `RecordId` dieser SQUEEZE Line highlighten.

9.1 Zwei mögliche Verknüpfungsstrategien

Es gibt zwei realistische Wege, eine Einkaufszeile wieder auf eine SQUEEZE Line zu mappen:

1. **Expliziter Allocation-Link** (nur in manchen Szenarien verfügbar)
   - Verwendet Felder auf `"DXP SQZ Document Line"`:
     - `"Allocated Document Type"` (enum `"DXP Order Match Document Type"`)
     - `"Allocated Document No."`
     - `"Allocated Document Line No."`
2. **Heuristischer Match nach Inhalt** (empfohlen als Fallback)
   - Verwendet Felder, die typischerweise auf beiden Seiten existieren (Type/No/Quantity/Unit Cost/Line Amount/Description).
   - Das funktioniert auch dann, wenn keine Allocation-Beziehung gepflegt wird.
   - Es ist nicht mathematisch perfekt (Duplikate sind möglich), daher sollten Matching-Regeln auf Ihren Use Case angepasst werden.

9.2 Empfohlenes Wiring (Host?Page übergibt SQZ?Kontext + Resident)

Sie benötigen in der Purchase-Lines-Subpage zwei Dinge:

1. Den Viewer-Resident (damit `PostMessage(...)` die richtige Control-Add-in Instanz trifft)
2. Den aktuellen SQZ Header-Kontext (damit Sie SQZ Lines auf das richtige Dokument filtern)

Empfohlenes Pattern auf der Host-Page (nachdem Sie `SqzHeader` aufgelöst haben):

```
trigger OnAfterGetCurrRecord()
var
    SharedResident: Codeunit "DXP SQUEEZE Viewer Resident";
    SqzHeader: Record "DXP SQZ Document Header";
begin
    if not TryResolveSqzHeaderForCurrentRecord(SqzHeader) then
        exit;

    CurrPage.DXPSQZViewer.Page.SetSqzRecord(SqzHeader);

    CurrPage.DXPSQZViewer.Page.GetViewerResident(SharedResident);
    ViewerCtx.SetResident(SharedResident);
    ViewerCtx.SetSqzContextForPurchInv(Rec."No.", SqzHeader."No.", SqzHeader."API Document
ID");
end;
```

9.3 Minimales Beispiel: Purch. Invoice Subform triggert Highlighting im Viewer

Dieses Beispiel erweitert **Purch. Invoice Subform**. Beachten Sie: der Datensatz der Basis-Subform ist ein `Record "Purchase Line"`, daher müssen die Resolver-Prozeduren `Record "Purchase Line"` akzeptieren.

Das hier gezeigte Allocation-Mapping ist die funktionierende Implementierung für Purchase-Invoice-Szenarien: es verknüpft über **Receipt** (`"Receipt No."` / `"Receipt Line No."`) statt über Belegnummer/Zeilennummer der Rechnung.

```
pageextension 50101 "MY Purch. Inv. Sf Mark" extends "Purch. Invoice Subform"
{
    trigger OnAfterGetCurrRecord()
```

```
    begin
        MarkSqzLineForPurchInvLine();
    end;

    var
        ApiMgt: Codeunit "DXP SQZ API Mgt.";
        Resident: Codeunit "DXP SQUEEZE Viewer Resident";
        ViewerState: Codeunit "DXP Viewer State";
        LineCoordMgt: Codeunit "DXP SQZ Coordinates Mgt.";
        ViewerCtx: Codeunit "DXP SQZ Purch. Inv. Viewer Ctx";
        SqzDocNo: Code[20];
        ApiDocumentId: Text[50];

    local procedure MarkSqzLineForPurchInvLine()
    var
        SqzLine: Record "DXP SQZ Document Line";
    begin
        if not ViewerCtx.TryGetSqzContextForPurchInv(Rec."Document No.", SqzDocNo,
ApiDocumentId) then
            exit;

        if not ViewerCtx.TryGetResident(Resident) then
            exit;

        if not TryResolveSqzLineForPurchLine(SqzDocNo, Rec, SqzLine) then
            exit;

        if not LineCoordMgt.CoordinateExists(SqzLine.RecordId) then
            exit;

        if ViewerState.GetViewerMode() = "DXP SQZ Viewer Mode"::Detached then
            ViewerState.UpdateViewerStateWithCoordinates(ApiDocumentId, SqzLine.RecordId)
        else
            ApiMgt.PostMessage(ApiMgt.MarkFieldByCoordinatesObj(SqzLine.RecordId), Resident);
    end;

    local procedure TryResolveSqzLineForPurchLine(SqzDocNo: Code[20]; PurchLine: Record
"Purchase Line"; var SqzLine: Record "DXP SQZ Document Line"): Boolean
    begin
        // 1) Allocation link via receipt (Purchase Invoice Szenario)
        SqzLine.Reset();
```

```
        SqzLine.SetRange("Document No.", SqzDocNo);
        SqzLine.SetFilter("Allocated Document No.", '%1&<>%2', PurchLine."Receipt No.", '');
        SqzLine.SetFilter("Allocated Document Line No.", '%1&<>%2', PurchLine."Receipt Line
No.", 0);
        if SqzLine.FindFirst() then
            exit(true);

        // 2) Fallback: content-based match (Type/No/Amounts)
        exit(TryResolveSqzLineByContent(SqzDocNo, PurchLine, SqzLine));
    end;

    local procedure TryResolveSqzLineByContent(SqzDocNo: Code[20]; PurchLine: Record "Purchase
Line"; var BestSqzLine: Record "DXP SQZ Document Line"): Boolean
    var
        Candidate: Record "DXP SQZ Document Line";
        BestScore: Integer;
        Score: Integer;
        QtyTol: Decimal;
        AmtTol: Decimal;
    begin
        QtyTol := 0.00001;
        AmtTol := 0.01;

        BestScore := 0;
        Candidate.Reset();
        Candidate.SetRange("Document No.", SqzDocNo);

        // Prefer narrowing early on stable fields
        Candidate.SetRange(Type, PurchLine.Type);
        if PurchLine."No." <> '' then
            Candidate.SetRange("No.", PurchLine."No.");

        if not Candidate.FindSet() then
            exit(false);

        repeat
            Score := 0;

            // Strong signals
            if Candidate.Type = PurchLine.Type then
                Score += 20;
```

```
            if (PurchLine."No." <> '') and (Candidate."No." = PurchLine."No.") then
                Score += 40;

            // Weak/medium signals (tolerant)
            if (Candidate.Quantity <> 0) and (Abs(Candidate.Quantity - PurchLine.Quantity) <=
QtyTol) then
                Score += 10;
            if (Candidate."Direct Unit Cost" <> 0) and (Abs(Candidate."Direct Unit Cost" -
PurchLine."Direct Unit Cost") <= AmtTol) then
                Score += 10;
            if (Candidate."Line Amount" <> 0) and (Abs(Candidate."Line Amount" -
PurchLine."Line Amount") <= AmtTol) then
                Score += 10;
            if (Candidate.Description <> '') and (PurchLine.Description <> '') and
(Candidate.Description = PurchLine.Description) then
                Score += 5;

            if Score > BestScore then begin
                BestScore := Score;
                BestSqzLine := Candidate;
            end;
        until Candidate.Next() = 0;

        // Guard: avoid picking a random line when nothing matches well.
        exit(BestScore >= 30);
    end;
}
```

Revision #1
Created 2026-01-29 09:16:13 UTC by Christoph Koepke-von Seth
Updated 2026-01-29 09:17:14 UTC by Christoph Koepke-von Seth