Benutzerdefinierte Aktionen: Übergabe des Quelldatensatzes an eigene Codeunits

Advanced Workflow kann an drei Stellen im Leben eines Workflows eine **eigene Codeunit** ausführen: bei Abschluss des Workflows, beim Abbruch mit Aktion und wenn eine Eskalationsstufe greift. In allen drei Fällen kann die App Ihrer Codeunit den **Quelldatensatz** übergeben, auf dem der Workflow läuft — die Einkaufsbestellung, den Debitor, den Artikel, also die Primärtabelle der Vorlage.

Diese Seite beschreibt den Entwicklervertrag: was genau übergeben wird, wie Sie es entgegennehmen und was die App mit dem Ergebnis macht.

Wo eine eigene Codeunit konfiguriert wird

Zeitpunkt	Konfiguration	Einstellung für die Übergabe
Workflow abgeschlossen	Karte **Workflowvorlage**, Abschnitt **Workflowaktion** → **Aktion bei Abschluss** = **Benutzerdefinierte Aktion**	**Quelldatensatz übergeben**
Workflow mit Aktion abgebrochen	Karte **Workflowvorlage**, Abschnitt **Konfiguration der benutzerdefinierten Abbruchaktion** → **Abbruchaktion** = **Benutzerdefinierte Aktion**	**Quelldatensatz bei Abbruch übergeben**
Eskalationsstufe erreicht	Karte **Eskalationsstufe** → Eskalationsaktion **Benutzerdefinierte Aktion**	**Quelldatensatz übergeben**

An allen drei Stellen wählen Sie **Objekttyp für benutzerdefinierte Aktion** = **Codeunit** und tragen die Objekt-ID in **Objekt-ID für benutzerdefinierte Aktion** ein. Das Nachschlagen ist auf Codeunits gefiltert, Sie können also aus der Liste auswählen, statt die Nummer zu tippen.

>

> Dieselben drei Einstellungen akzeptieren als Objekttyp auch **Bericht**. Ein Bericht wird dann als PDF gerendert, nicht ausgeführt; siehe*Übergabe an einen Bericht* am Ende dieser Seite.

Was genau übergeben wird

Ist **Quelldatensatz übergeben** aktiviert, ruft die App Ihre Codeunit mit dem Quelldatensatz auf. In AL bedeutet das: der `OnRun`-Trigger Ihrer Codeunit erhält diesen Datensatz in `Rec`. Ist die Einstellung deaktiviert, wird die Codeunit **ohne** Datensatz ausgeführt — `Rec` ist dann nicht gefüllt.

Zwei Eigenschaften der Übergabe sind in der Praxis wichtig:

- **Es wird genau ein Datensatz übergeben, und es sind keine Filter gesetzt.** Ihre Codeunit erhält den einzelnen Quellbeleg. Benötigt Ihre Logik die zugehörigen Zeilen, lesen Sie diese selbst über die Schlüsselfelder des Kopfes.
- **Übergeben wird der Quellbeleg, auf dem der Workflow gestartet wurde, identifiziert über seine Datensatz-ID.** Vor einer Abschlussaktion liest die App den Datensatz neu aus der Datenbank, wenn er noch nicht gespeicherte Änderungen im Speicher hat — Ihr Code sieht also den gespeicherten Zustand und keine halb geänderte Kopie. Hängt Ihre Logik an einem Feld, das andere Prozesse zwischenzeitlich geändert haben können, kostet es nichts, den Datensatz am Anfang von `OnRun` mit `Get` erneut zu lesen und sicher zu sein.

Die Codeunit schreiben

Deklarieren Sie `TableNo` mit der Tabelle, auf der Ihre Workflowvorlage läuft. Erst dadurch ist `Rec` ein typisierter Datensatz, den Sie über Feldnamen ansprechen können:

```
codeunit 50100 "MY Release Approved PO"
{
    TableNo = "Purchase Header";

    trigger OnRun()
    var
        MySetup: Record "MY Integration Setup";
    begin
        // Rec ist die Einkaufsbestellung, auf der der Workflow lief.
        Rec.TestField("Document Type", Rec."Document Type"::Order);

        MySetup.Get();
        if not MySetup."Notify ERP Gateway" then
```

```
            exit;

        SendToGateway(Rec."No.", Rec."Buy-from Vendor No.", Rec.Amount);
    end;

    local procedure SendToGateway(DocumentNo: Code[20]; VendorNo: Code[20]; DocAmount:
Decimal)
    begin
        // Ihr Integrationsaufruf
    end;
}
```

Tragen Sie in der Vorlage unter **Objekt-ID für benutzerdefinierte Aktion** die `50100` ein, lassen Sie **Quelldatensatz übergeben** aktiviert — und die Codeunit läuft bei jedem Abschluss eines Workflows auf **Einkaufskopf**.

Den Quelldatensatz ändern

Sie dürfen den Quelldatensatz in Ihrer Codeunit ändern und speichern. Die App unterdrückt während der Aktion ihre eigene Auslösererkennung, sodass Ihr `Modify` keinen zweiten Workflow auf demselben Beleg startet:

```
    trigger OnRun()
    begin
        Rec."Your Custom Status" := Rec."Your Custom Status"::Approved;
        Rec."Approved On" := CurrentDateTime();
        Rec.Modify(true);
    end;
```

> Wenn Sie lediglich einige Felder auf feste Werte setzen möchten, brauchen Sie gar keine Codeunit — wählen Sie stattdessen die Aktion **Feldwertzuweisungen festlegen** und konfigurieren Sie die Felder im Abschnitt **Feldwertzuweisungen**.

Mehrere Tabellen bedienen

Da `TableNo` pro Codeunit fest ist, passt eine Codeunit zu den Vorlagen einer Tabelle. Soll dieselbe Logik Vorlagen mehrerer Tabellen bedienen, schreiben Sie je Tabelle eine schlanke Codeunit und halten die Logik in einer gemeinsamen Codeunit, die alle aufrufen:

```
codeunit 50101 "MY Workflow Action - Purch"
{
    TableNo = "Purchase Header";

    trigger OnRun()
    var
        SharedLogic: Codeunit "MY Workflow Action Logic";
    begin
        SharedLogic.NotifyGateway(Rec.RecordId(), Rec."No.", Rec.Amount);
    end;
}

codeunit 50102 "MY Workflow Action - Sales"
{
    TableNo = "Sales Header";

    trigger OnRun()
    var
        SharedLogic: Codeunit "MY Workflow Action Logic";
    begin
        SharedLogic.NotifyGateway(Rec.RecordId(), Rec."No.", Rec.Amount);
    end;
}
```

Jede Vorlage verweist dann auf die Codeunit, die zu ihrer eigenen Primärtabelle passt.

> Geben Sie der Codeunit das `TableNo` der Primärtabelle der Vorlage. Passen die beiden nicht zusammen, kann der Datensatz nicht übergeben werden und der Lauf wird im Audit-Protokoll als fehlgeschlagen vermerkt. Eine Codeunit ganz ohne `TableNo` ist nur sinnvoll, wenn **Quelldatensatz übergeben** deaktiviert ist.

Fehlerbehandlung und Audit-Protokoll

Jeder Lauf wird im Audit-Protokoll des Workflows vermerkt, das Sie über die Workflowinstanz finden (siehe **Workflowinstanzen finden**):

- Bei Erfolg — **Benutzerdefinierte Codeunit 50100 erfolgreich ausgeführt**
- Bei Fehlschlag — **Ausführung der benutzerdefinierten Codeunit 50100 fehlgeschlagen**

- Wenn Objekttyp oder ID fehlen — **Benutzerdefinierte Aktion nicht richtig konfiguriert**

> Ein Fehler in Ihrer Codeunit setzt den Workflow **nicht** zurück und verhindert den Abschluss des Belegs nicht. Der Workflow bleibt abgeschlossen, der Fehlschlag wird ins Audit-Protokoll geschrieben, und die Verarbeitung läuft weiter. Darf Ihre Integration nicht stillschweigend verloren gehen, lassen Sie Ihre Codeunit den Fehlschlag selbst festhalten — etwa als Warteschlangeneintrag oder Protokollsatz, den Sie überwachen — statt sich darauf zu verlassen, dass der Fehler beim Anwender ankommt.

Aus demselben Grund gehören `Message` oder `Confirm` nicht in eine benutzerdefinierte Aktion. Die Aktion läuft häufig in einer Hintergrundsitzung — bei Eskalation aus der Aufgabenwarteschlange, bei Abschluss gegebenenfalls als Hintergrundaufgabe — wo niemand einen Dialog beantworten kann.

Übergabe an einen Bericht

Bei **Objekttyp für benutzerdefinierte Aktion** = **Bericht** öffnet die App keine Anforderungsseite. Sie rendert den Bericht im Hintergrund als PDF und filtert bei aktiviertem **Quelldatensatz übergeben** den Datenbestand des Berichts vorher auf genau diesen Quelldatensatz. Ihr Bericht braucht dafür ein `DataItem` auf der Quelltabelle, damit der Filter greifen kann.

> Das erzeugte PDF wird nicht automatisch gespeichert. Soll das Dokument archiviert werden, verwenden Sie eine Codeunit, die es erzeugt und ablegt, oder aktivieren Sie in der Vorlage **Audit-Trail in Anhänge speichern** für das Audit-Trail-PDF des Workflows selbst.

Verwandte Seiten

- **Workflowaktion bei Abschluss** — die Aktion in der Vorlage konfigurieren
- **Modul: Eskalations-Engine** — Eskalationsstufen und ihre Aktionen
- **Workflowinstanzen finden** — wo das Audit-Protokoll zu lesen ist

---

Revision #1
Created 2026-08-18 10:44:20 UTC by Bernd Feddersen
Updated 2026-08-18 10:44:20 UTC by Bernd Feddersen