



T270 Workflow APIs

Varthini Bhaskaran

Technical Account Manager

Timing and Agenda

October 31, 2022 -10 -11 AM

Day 1

Lesson 1.1: Set Up the Basic Parts of the Workflow

November 1, 2022 -10 -11 AM

Day 2

Lesson 1.2: Implement a Simple Transition

Lesson 1.3: Implement a Group of Transitions

Lesson 1.4: Implement a Transition with a Dialog Box

Timing and Agenda

November 2, 2022 -10 -11 AM

Day 3

Lesson 1.5: Implement a Transition with Field Assignments

Lesson 2.1: Use an Existing Event in a Custom Workflow

November 3, 2022 -10 -11 AM

Day 4

Lesson 2.2: Create a New Event

Lesson 3.1: Customize the Workflow for Invoices

Day 1



Part 1: Creating a Custom Workflow

Lesson 1.1: Set Up the Basic Parts of the Workflow

Learning Objectives

In this lesson, you will learn how to do the following:

- Determine the set of states of a document in the workflow
- Define the set of states for the workflow
- Create the screen configuration method
- Enable workflow validations

Lesson Summary

In this lesson, you have prepared the base components of the workflow. You have learned how to create a workflow class, add a screen configuration, and define the set of states for the workflow.

Day 2

Lesson 1.2: Implement a Simple Transition

Learning Objectives

In this lesson, you will learn how to do the following:

- Define and configure workflow actions
- Define a default category for workflow actions
- Define and configure workflow states
- Define a simple transition

Figure: A new document on the Repair Work Orders form

Repair Work Orders

Battery Replacement

NOTES FILES CUSTOMIZATION TOOLS ▾

← ↻ 📁 ↶ + 🗑️ 📄 ▾ |< < > >| REMOVE HOLD

Order Nbr.: <NEW> 🔍

Status: On Hold

* Date Created: 12/23/2021 ▾

Date Completed:

Priority: Medium ▾

* Customer ID: C000000001 - Jersey Central Office E 🔍

* Service: BATTERYREPLACE - Battery Replace 🔍

* Device: NOKIA3310 - Nokia 3310 🔍

Assignee:

Description: Battery replacement, Nokia 3310

Order Total: 35.00

Invoice Nbr.:

REPAIR ITEMS LABOR

↻ + ✎ ✕ |↔️ 🗑️

📄 🔍	Repair Item Type	Inventory ID	Description	Price
> 🔍	Battery	BAT3310	Battery for Nokia 3310	20.00
🔍	Back Cover	BCOV3310	Back cover for Nokia 3310	10.00

|< < > >|

Figure: A document with the Ready for Assignment status

Repair Work Orders

000003 - Battery Replacement

NOTES FILES CUSTOMIZATION TOOLS ▾

← ↻ 📄 ↶ + 🗑️ 📋 ▾ ⏪ < > ⏩ ...

Order Nbr.:	000003 🔍	Customer ID:	C000000001 - Jersey Central Office Equip	Order Total:	35.00
Status:	Ready for Assignment	Service:	BATTERYREPLACE - Battery Replacement	Invoice Nbr.:	
* Date Created:	12/23/2021 ▾	Device:	NOKIA3310 - Nokia 3310		
Date Completed:		Assignee:	🔍		
Priority:	Medium ▾	Description:	Battery replacement, Nokia 3310		

REPAIR ITEMS LABOR

↻ + ✎ ✕ ⏪ ⏩

📋	🔍	📄	Repair Item Type	Inventory ID	Description	Price
>	🔍	📄	Battery	BAT3310	Battery for Nokia 3310	20.00
	🔍	📄	Back Cover	BCOV3310	Back cover for Nokia 3310	10.00

⏪ < > ⏩

Lesson Summary

In this lesson, you have implemented the first transition for the workflow of the Repair Work Orders (RS301000) form. You have learned how to define workflow actions, workflow states, and a transition. You have also tested the transition.

Lesson 1.3: Implement a Group of Transitions

Learning Objectives

In this lesson, you will learn how to do the following:

- Define a condition in a workflow
- Specify a condition for a transition
- Unite transitions into a group

Lesson Summary

In this lesson, you have implemented a transition from the OnHold state to the PendingPayment state and added conditions for previously implemented transitions. You have learned how to put transitions into groups and declare condition packs.

Exercise 1.1: Implement the PutOnHold Transition

Repair Work Orders

000003 - Battery Replacement

NOTES FILES CUSTOMIZATION TOOLS

← ↻ ↺ + 🗑️ 📄 < > >| HOLD ...

Order Nbr.: 000003 Customer ID: C000000001 - Jersey C 35.00

Status: Ready for Assignment Service: BATTERYREPLACE - 1

* Date Created: 12/23/2021 Device: NOKIA3310 - Nokia 3310

Date Completed: Assignee:

Priority: Medium Description: Battery replacement, Nokia 3310

REPAIR ITEMS LABOR

↻ + ✎ × |↔| ☒

Repair Item Type	Inventory ID	Description	Price
Battery	BAT3310	Battery for Nokia 3310	20.00
Back Cover	BCOV3310	Back cover for Nokia 3310	10.00

Figure: The More menu of the Repair Work Orders form

Lesson 1.4: Implement a Transition with a Dialog Box

Learning Objectives

In this lesson, you will learn how to do the following:

- Define a dialog box in a workflow
- Specify a dialog box for an action

Figure: The Assign button

Repair Work Orders

000003 - Battery Replacement

NOTES FILES CUSTOMIZATION TOOLS ▾

← ↻ 📄 ↶ + 🗑️ 📄 ▾ |< < > >| HOLD **ASSIGN** ⋮

Order Nbr.: 000003 🔍

Status: Ready for Assignment

* Date Created: 12/23/2021 ▾

Date Completed:

Priority: Medium ▾

Customer ID: C000000001 - Jersey Central Office Equi

Service: BATTERYREPLACE - Battery Replaceme

Device: NOKIA3310 - Nokia 3310

Assignee: 🔍

Description: Battery replacement, Nokia 3310

Order Total: 35.00

Invoice Nbr.:

REPAIR ITEMS LABOR

🔄 + ✎ ✕ ⏮ ⏭

🔍	📄	📄	Repair Item Type	Inventory ID	Description	Price
>	🔍	📄	Battery	BAT3310	Battery for Nokia 3310	20.00
	🔍	📄	Back Cover	BCOV3310	Back cover for Nokia 3310	10.00

|< < > >|

Lesson Summary

In this lesson, you have implemented the Assign action, a transition from the ReadyForAssignment state to the Assigned state, and a dialog box that is shown when a user clicks the Assign button on the form toolbar.

Day 3

Lesson 1.5: Implement a Transition with Field Assignments

Learning Objectives

In this lesson, you will learn how to do the following:

- Define a DAC field assignment with the workflow API methods
- Define the location of the button and command associated with a workflow action

Lesson Summary

In this lesson, you have implemented the Complete action, as well as its associated button on the table toolbar and command on the More menu. You have also defined the transition from the Assigned state to the Completed state.

Repair Work Orders

[CREATE INVOICE](#)

NOT

Executing. Press to abort
00:00:01

CANCEL

Status: Completed

* Date Created: 12/23/2021

Date Completed: 12/24/2021

Priority: Medium

Customer ID: C000000001 - Jersey Central Office Equi

Service: BATTERYREPLACE - Battery Replacement

Device: NOKIA3310 - Nokia 3310

Assignee: Andrews, Michael

Description: Battery replacement, Nokia 3310

Order Total: 35.00

Invoice Nbr.:

REPAIR ITEMS

LABOR


↺ + ↻ × |↔| ☒

		Repair Item Type	Inventory ID	Description	Price
>		Battery	BAT3310	Battery for Nokia 3310	20.00
		Back Cover	BCOV3310	Back cover for Nokia 3310	10.00

Figure: The invoice number of the repair work order

Repair Work Orders

000003 - Battery Replacement

NOT  The operation has completed.

Order Nbr.: 000003 Customer ID: C000000001 - Jersey Central Office Equi Order Total: 35.00

Status: Completed Service: BATTERYREPLACE - Battery Replaceme Invoice Nbr.: **INV000049**

* Date Created: 12/23/2021 Device: NOKIA3310 - Nokia 3310

Date Completed: 12/24/2021 Assignee: Andrews, Michael

Priority: Medium Description: Battery replacement, Nokia 3310

REPAIR ITEMS LABOR

Repair Item Type	Inventory ID	Description	Price
Battery	BAT3310	Battery for Nokia 3310	20.00
Back Cover	BCOV3310	Back cover for Nokia 3310	10.00

Part 1 Summary

In the part of the course, you have learned how to define a screen configuration. You have learned how to define workflow states, actions, dialog boxes, and transitions. You have also learned how to configure these parts of the screen configuration.



Part 2: Incorporating an Existing Workflow into a Custom Workflow

Lesson 2.1: Use an Existing Event in a Custom Workflow

Learning Objectives

In this lesson, you will learn how to do the following:

- Explore the code of the predefined workflow to find the event
- Create a custom event handler
- Bind the event handler to an event
- Register an event handler in a particular state
- Implement a transition triggered by an event
- Override the Persist method

Figure: The cleared the Enable Just My Code check box

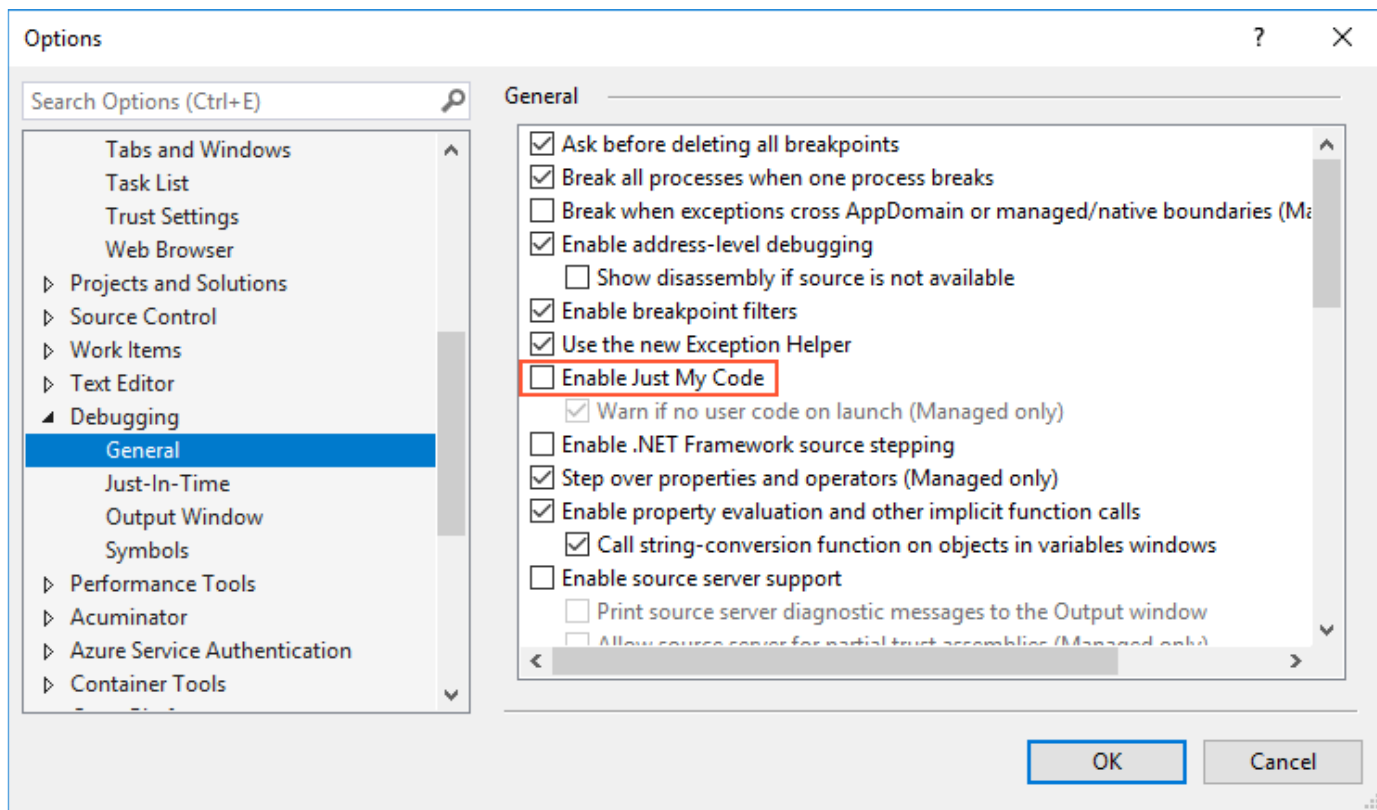
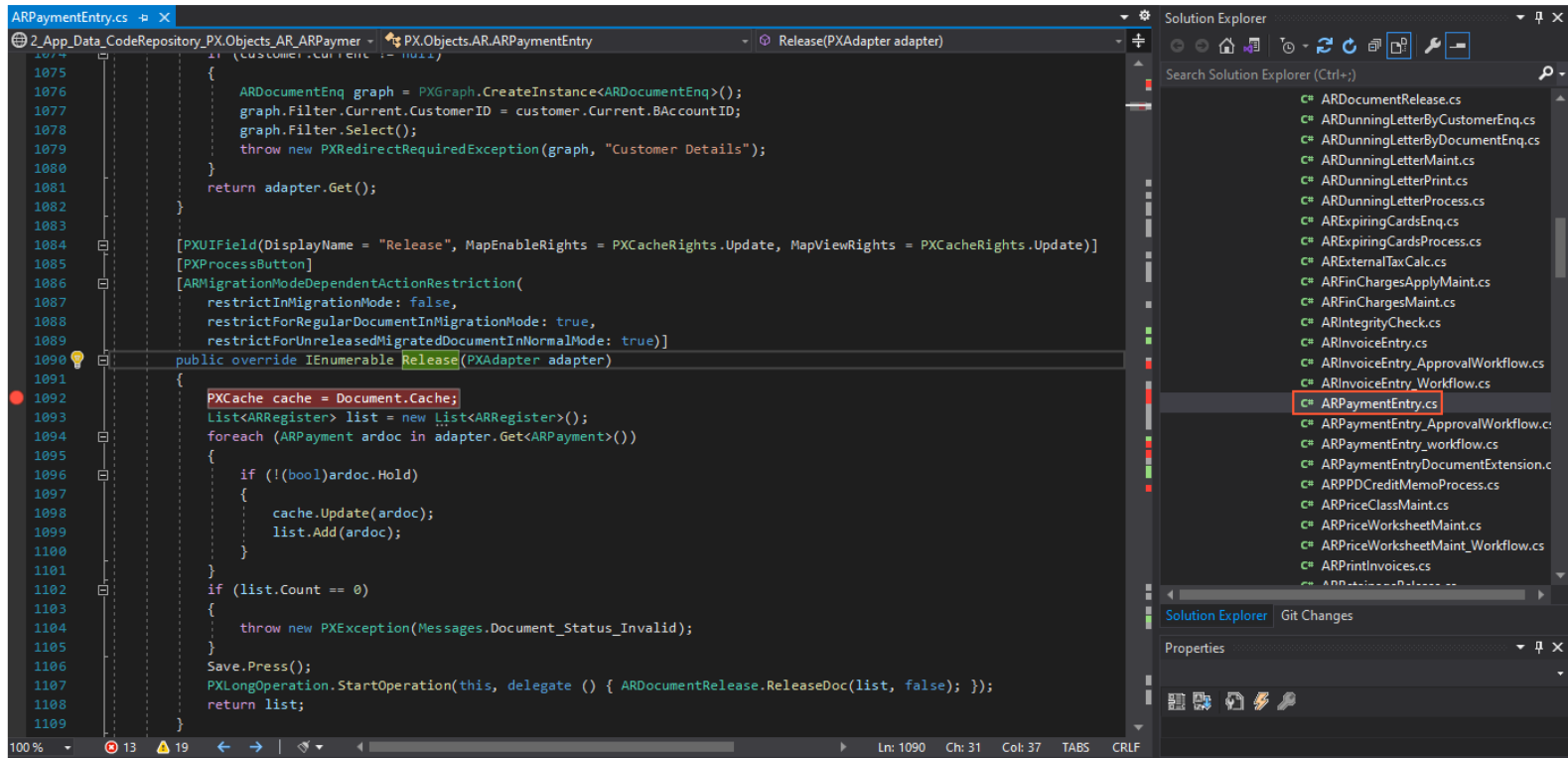


Figure: The source code of the Release action



```
ARPaymentEntry.cs
2_App_Data_CodeRepository.PX.Objects_AR_ARPaymentEntry
Release(PXAdapter adapter)

1074
1075 {
1076     ARDocumentEnq graph = PXGraph.CreateInstance<ARDocumentEnq>();
1077     graph.Filter.Current.CustomerID = customer.Current.BAccountID;
1078     graph.Filter.Select();
1079     throw new PXRedirectRequiredException(graph, "Customer Details");
1080 }
1081 return adapter.Get();
1082 }
1083
1084 [PXUIField(DisplayName = "Release", MapEnableRights = PXCacheRights.Update, MapViewRights = PXCacheRights.Update)]
1085 [PXProcessButton]
1086 [ARMigrationModeDependentActionRestriction(
1087     restrictInMigrationMode: false,
1088     restrictForRegularDocumentInMigrationMode: true,
1089     restrictForUnreleasedMigratedDocumentInNormalMode: true)]
1090 public override IEnumerable Release(PXAdapter adapter)
1091 {
1092     PXCache cache = Document.Cache;
1093     List<ARRegister> list = new List<ARRegister>();
1094     foreach (ARPayment ardoc in adapter.Get<ARPayment>())
1095     {
1096         if (!(bool)ardoc.Hold)
1097         {
1098             cache.Update(ardoc);
1099             list.Add(ardoc);
1100         }
1101     }
1102     if (list.Count == 0)
1103     {
1104         throw new PXException(Messages.Document_Status_Invalid);
1105     }
1106     Save.Press();
1107     PXLongOperation.StartOperation(this, delegate () { ARDocumentRelease.ReleaseDoc(list, false); });
1108     return list;
1109 }
```

Solution Explorer

Search Solution Explorer (Ctrl+):

- ARDocumentRelease.cs
- ARDunningLetterByCustomerEnq.cs
- ARDunningLetterByDocumentEnq.cs
- ARDunningLetterMaint.cs
- ARDunningLetterPrint.cs
- ARDunningLetterProcess.cs
- ARExpiringCardsEnq.cs
- ARExpiringCardsProcess.cs
- ARExternalTaxCalc.cs
- ARFinChargesApplyMaint.cs
- ARFinChargesMaint.cs
- ARIntegrityCheck.cs
- ARInvoiceEntry.cs
- ARInvoiceEntry_ApprovalWorkflow.cs
- ARInvoiceEntry_Workflow.cs
- ARPaymentEntry.cs**
- ARPaymentEntry_ApprovalWorkflow.cs
- ARPaymentEntry_workflow.cs
- ARPaymentEntryDocumentExtension.c
- ARPPDCreditMemoProcess.cs
- ARPriceClassMaint.cs
- ARPriceWorksheetMaint.cs
- ARPriceWorksheetMaint_Workflow.cs
- ARPrintInvoices.cs
- ARPrintInvoicesRelease.cs

Solution Explorer Git Changes

Properties

100% 13 19 Ln: 1090 Ch: 31 Col: 37 TABS CRLF

Figure: The call hierarchy for the CloseInvoiceAndClearBalances method

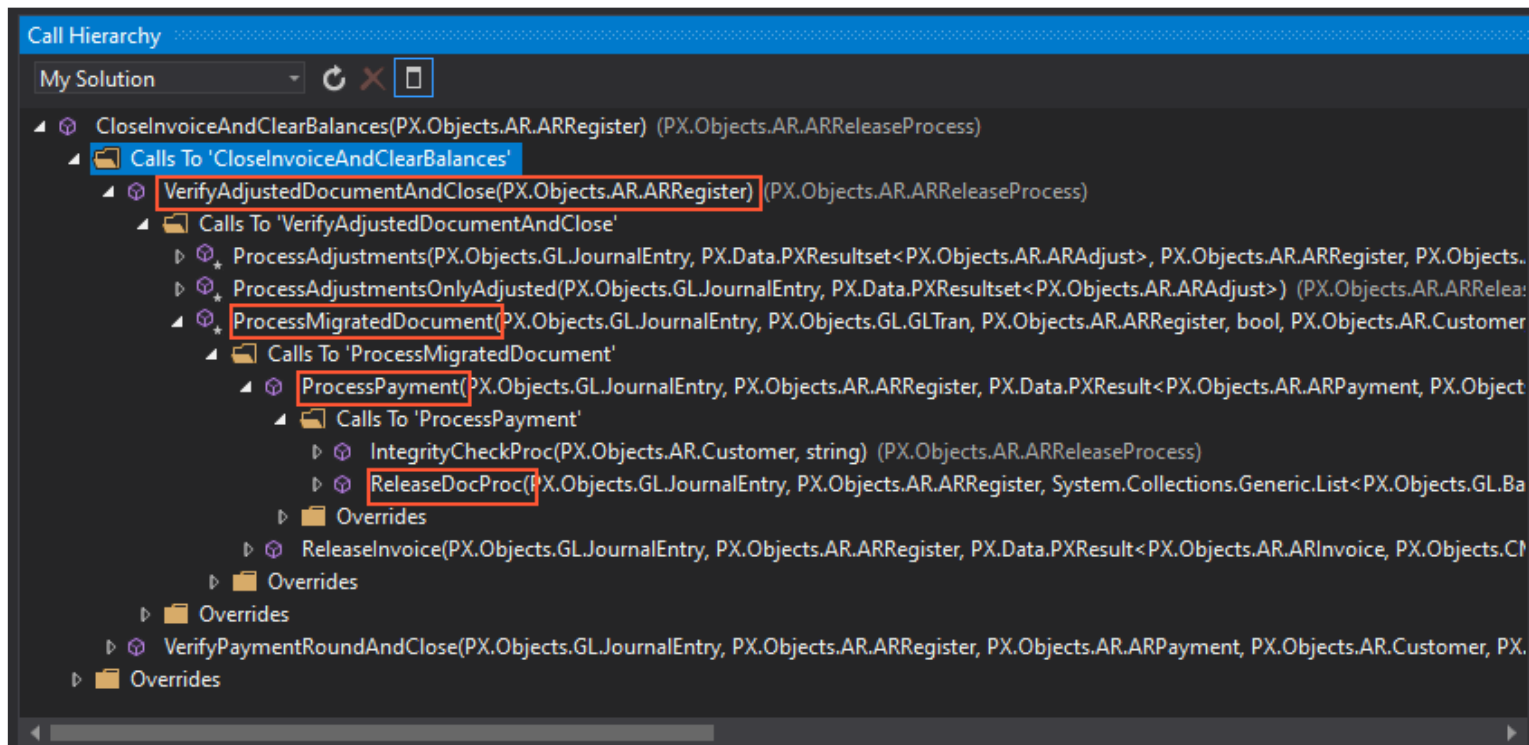


Figure: Transitions from the Open state

```
transitions.AddGroupFrom<State.open>(ts =>
{
    ts.Add(t => t
        .To<State.open>()
        .IsTriggeredOn(g=>g.emailInvoice)
        .WithFieldAssignments(fass => fass.Add<emailed>(v => v.SetFromValue(true))));
    ts.Add(t => t
        .To<State.closed>()
        .IsTriggeredOn(g => g.OnReleaseDocument)
        .When(conditions.IsClosed));
    ts.Add(t => t
        .To<State.closed>()
        .IsTriggeredOn(g => g.OnCloseDocument));
    ts.Add(t => t
        .To<State.canceled>()
        .IsTriggeredOn(g => g.OnCancelDocument));
});
```


Figure: The repair work order with the Paid status

Repair Work Orders

000005 - Battery Replacement

NOTES FILES CUSTOMIZATION TOOLS

← ↻ ↺ + 🗑️ 📄 |< < > >| ...

Order Nbr.: 000005 Customer ID: C000000001 - Jersey Central Office Equip Order Total: 35.00

Status: **Paid** Service: BATTERYREPLACE - Battery Replacement Invoice Nbr.: INV0000050

* Date Created: 12/28/2021 Device: NOKIA3310 - Nokia 3310

Date Completed: 12/28/2021 Assignee: Andrews, Michael

Priority: Medium Description: Battery replacement, Nokia 3310

REPAIR ITEMS LABOR

↻ + ✎ × |↔| 🗑️

📄	🔍	📄	Repair Item Type	Inventory ID	Description	Price
>	🔍	📄	Battery	BAT3310	Battery for Nokia 3310	20.00
	🔍	📄	Back Cover	BCOV3310	Back cover for Nokia 3310	10.00

|< < > >|

Lesson Summary

In this lesson, you have used the `OnCloseDocument` event to trigger the transition from the Completed state to the Paid state. You have implemented the Paid state, created a custom event handler, and registered the event handler in the screen configuration. You have also overridden the `Persist` method to save changes to the database.

Day 4

Lesson 2.2: Create a New Event

Learning Objectives

In this lesson, you will learn how to do the following:

- Derive the value for a custom field from another form
- Create a custom event
- Override a method to fire the event

Payments and Applications

NOTES ACTIVITIES FILES CUSTOMIZATION TOOLS

Type:	Payment	Customer:	C000000001 - Jersey Central Office Equip	Payment Amo...	50.00	Prepayment Percent:	5.00
Reference Nbr.:	<NEW>	Payment Meth...	CHECK - Check Payment	Applied to Doc...	3.00		
Status:	On Hold	Card/Account ...		Applied to Ord...	0.00		
* Application Date:	12/29/2021	* Cash Account:	102000-YOGI - Checking Account	Available Bala...	47.00		
* Application Pe...	12-2021			Write-Off Amo...	0.00		
Payment Ref.:	000003			Finance Charg...	0.00		
				Deducted Cha...	0.00		
		Description:					

DOCUMENTS TO APPLY

SALES ORDERS

APPLICATION HISTORY

FINANCIAL

APPROVALS

CHARGES



Acumatica
The Cloud ERP

Figure: The repair work order with the Ready for Assignment status

Repair Work Orders

000004 - Liquid Damage

NOTES FILES CUSTOMIZATION TOOLS ▾

← ↻ + 🗑️ 📄 ▾ |< < > >| HOLD **ASSIGN** ...

Order Nbr.: 000004 Customer ID: C000000001 - Jersey Central Office Equip Order Total: 50.00

Status: **Ready for Assignment** Service: LIQUIDDAMAGE - Liquid Damage Invoice Nbr.: INV000051

* Date Created: 12/23/2021 Device: NOKIA3310 - Nokia 3310

Date Completed: Assignee:

Priority: Medium Description: Liquid Damage, Nokia 3310

REPAIR ITEMS LABOR

↻ + ✎ × |↔️ ☒

📄	🔗	📄	Repair Item Type	Inventory ID	Description	Price
>	🔗	📄	Battery	BAT3310	Battery for Nokia 3310	20.00
	🔗	📄	Battery	BAT3310EX	Extended Battery for Nokia 3310	30.00

|< < > >|

Lesson Summary

In this lesson, you have learned how to create a custom workflow event. You have implemented a new event, defined an event handler for it, and fired the event in an overridden method. You have also learned how to derive values from one entity and copy them to another entity.

Part 2 Summary

In this part, you have learned how to define a custom workflow event and fire it and how to use existing workflow events in a custom workflow. You have also learned how to derive a value of a custom field from another entity and how to explore the source code of Acumatica ERP.

You have completed the implementation of the workflow for the Repair Work Orders (RS301000) form. To verify that all elements of the workflow have been implemented, you can open the state diagram for the Repair Work Orders form and compare it to the one shown in Customization Description.



Part 3: Customizing an Existing Workflow

Lesson 3.1: Customize the Workflow for Invoices

Learning Objectives

In this lesson, you will learn how to do the following:

- Update an existing screen configuration
- Define a new action category
- Define a workflow action based on a graph action

Lesson Summary

In this lesson, you have learned how to customize an existing workflow. You have implemented a new action and a new action category, and added them to the customized workflow.



Thank you

Varthini Bhaskaran