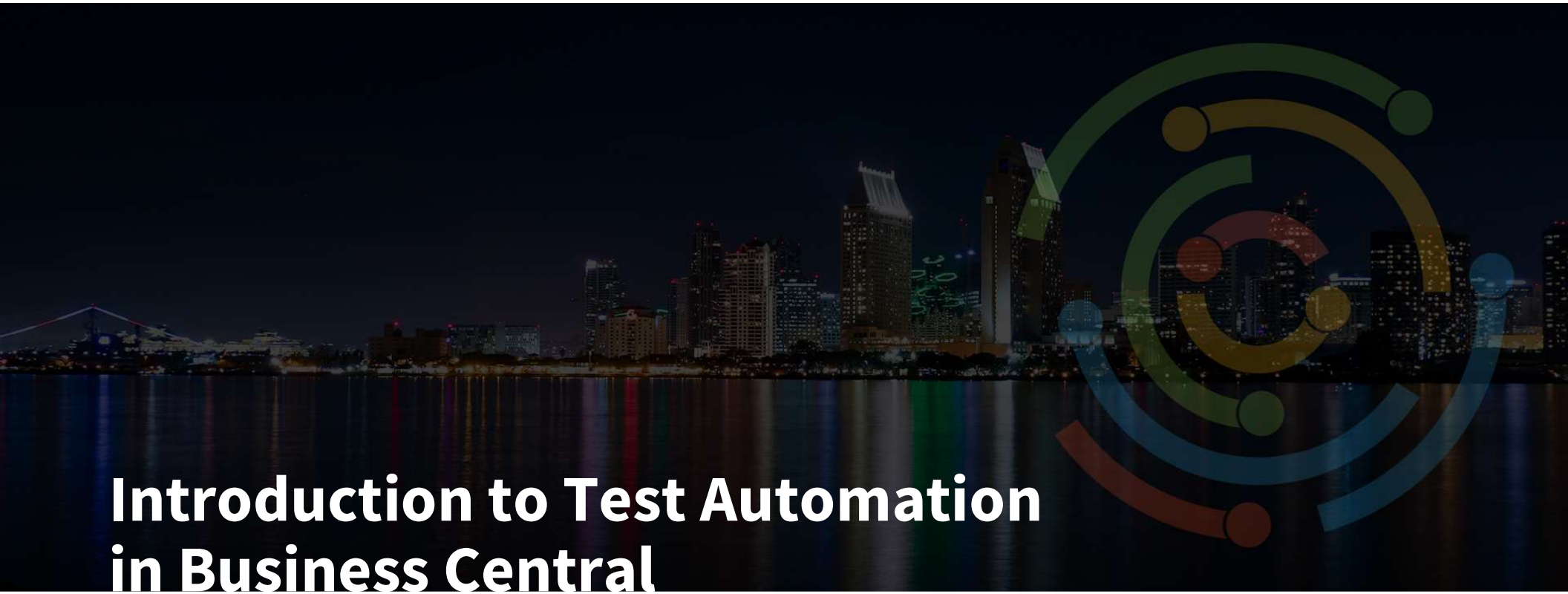


Introduction to Test Automation in Business Central

"Testing can show the presence of errors, but not their absence."

- Edsger W. Dijkstra

By Partners, For Partners
DIRECTIONS NORTH AMERICA

April 14-17, 2024 Sheraton San Diego Hotel & Marina, San Diego, CA

About Me

Brad Prendergast
Tigunia, LLC

brad@tigunia.com
@dvlprlife
linkedin.com/in/brad-prendergast/

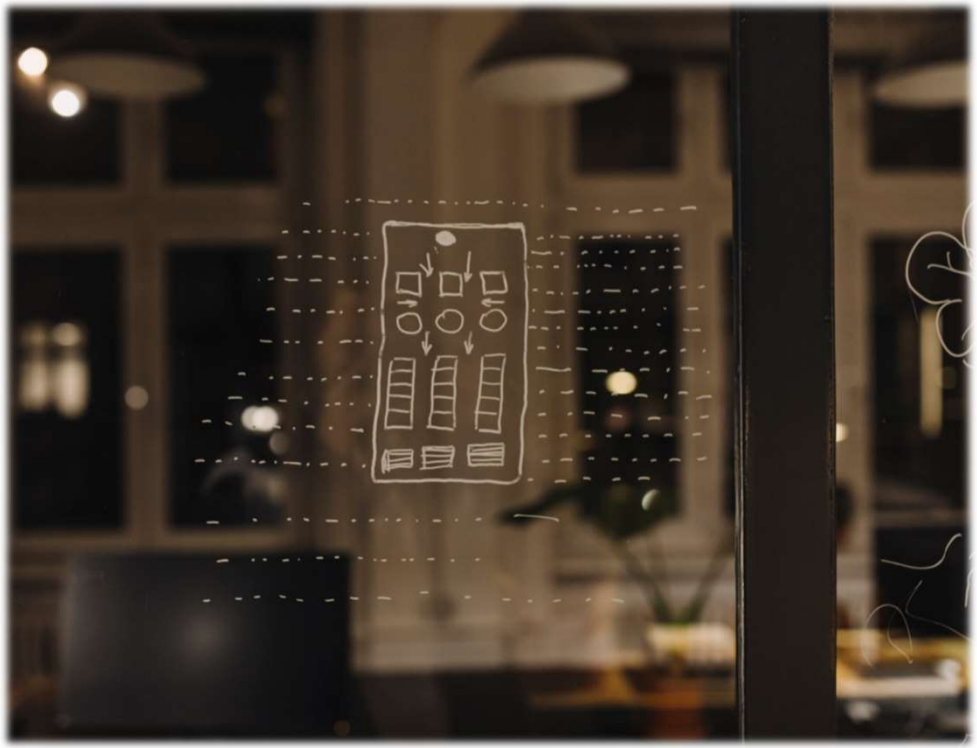
Microsoft® MVP Most Valuable Professional

By Partners, For Partners
DIRECTIONS NORTH AMERICA

April 14-17, 2024 Sheraton San Diego Hotel & Marina, San Diego, CA

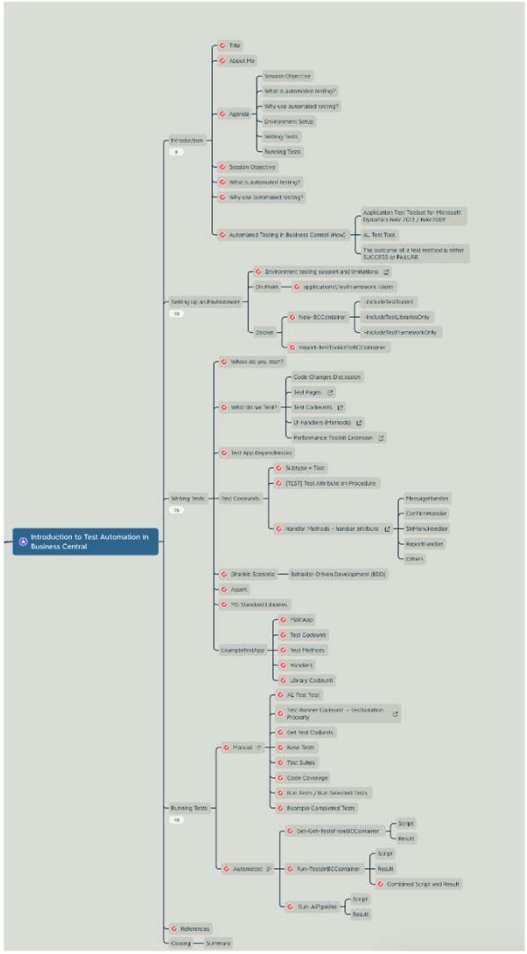
Agenda

- Objective
- What is automated testing?
- Why use automated testing?
- Setting up an Environment
- Writing Tests
- Running Tests



Session Objective

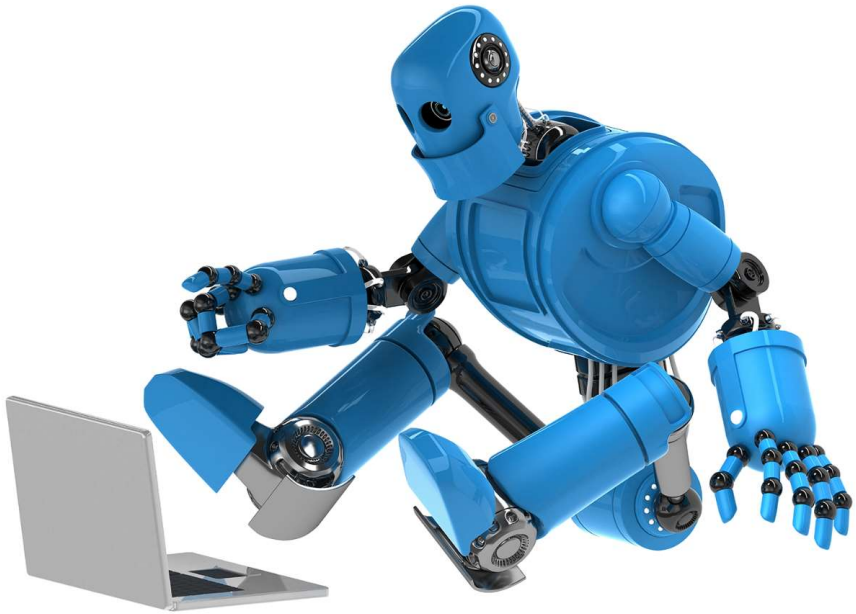
- Basic Understanding of Automated Testing in Business Central
  - Environment Setup
  - Writing Tests
  - Running Tests



What is Automated Testing

- Automated testing in Business Central involves using software to run tests automatically on the system to ensure that it functions as expected. This approach reduces manual testing, which can be time-consuming, error-prone, and costly.

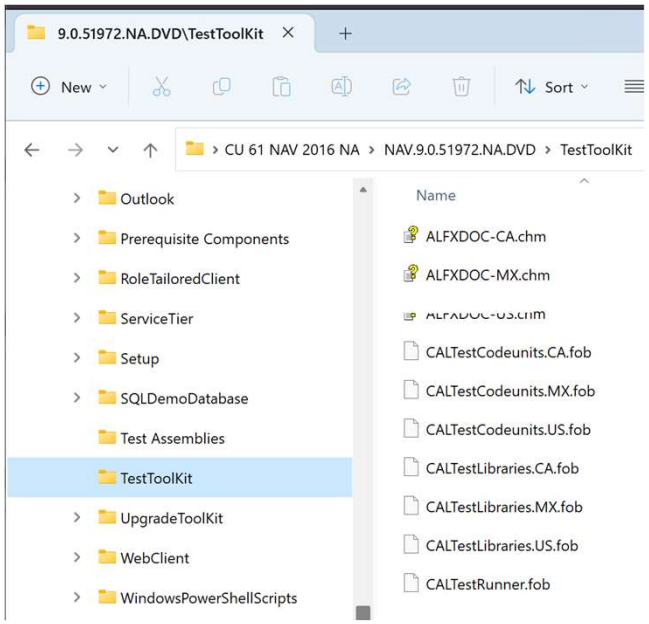
- Automated testing in Business Central involves creating a set of test cases that simulate different scenarios and use cases. These test cases are then executed automatically and checks the output against the expected results. The results of each test are analyzed to identify any defects or issues.

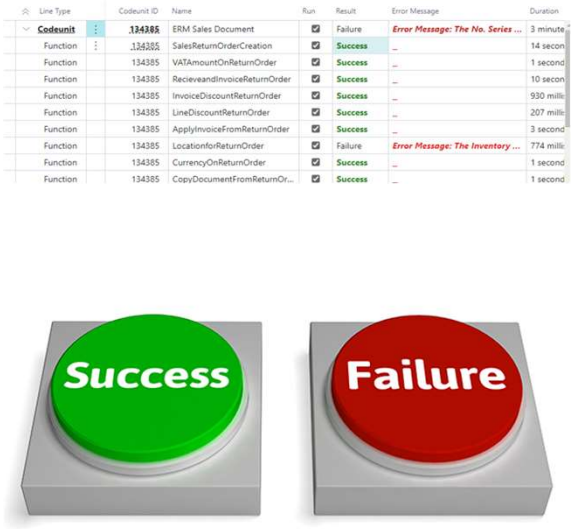


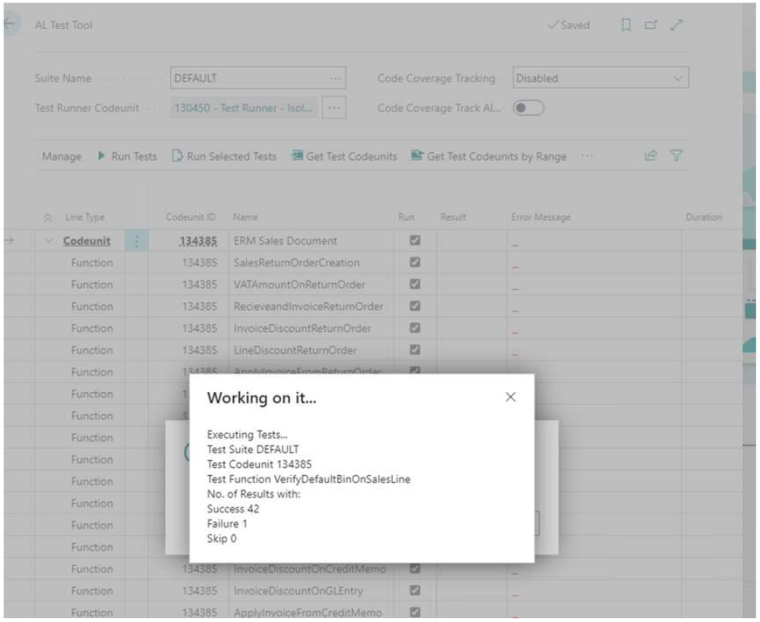
Why use Automated Testing

- Saves Time
- Improves Quality*
- Repeatable and Reliable
- Reduces Risk and Cost
- Less Disruption / Maximize Error Detection
- Latest Releases

Automated Testing in Business Central







Setting up an Environment

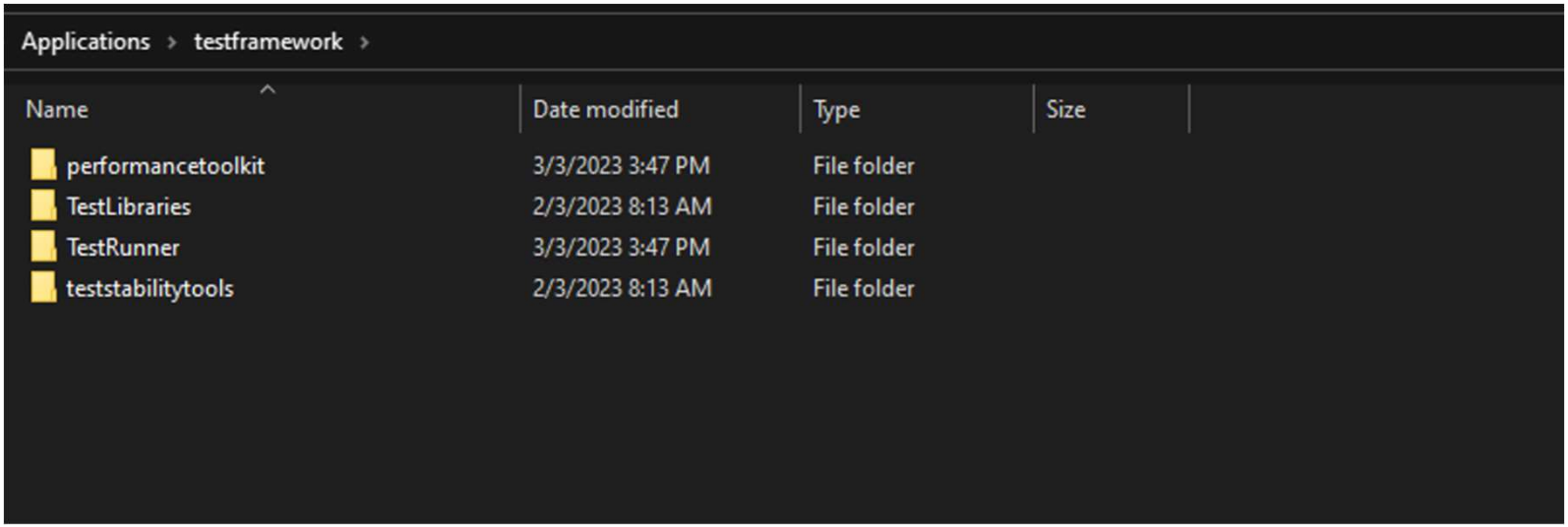
- Environment testing support and limitations

Business Central solution	Environment	Testing allowed	More details
Online	Production		Running tests isn't allowed because it might have an adverse effect on your business. Testing can incidentally invoke external systems, like CDS, PayPal, and web hook subscriptions. Invoking these systems may slow down the solution for other users or cause data corruption.
	Sandbox	✓	You can use a sandbox environment to run tests manually to verify functionality on an environment. Running a large number of tests or tests that take a long time (more than 15 minutes per test method) isn't allowed. It's recommended that you don't run tests more than one or two hours a day.
On-premises	Production	✓	For Business Central on-premises, running automated tests is only possible with a Partner license or a license that includes the Application Builder module. You can disable the ability to run tests by turning off **Enable Test Automation** (TestAutomationEnabled) on the Business Central Server instance. For more information, see Configuring Business Central Server - General Settings.
	Container-based development environment	✓	This setup should be the default environment for running large number of tests or setting up CI/CD gates. For more information, see Running a Container-Based Development Environment or Running Tests In Containers.

Source https://learn.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/devenv-testing-application

Setting up an Environment

- Available on Microsoft Dynamics 365 Business Central installation “DVD”



Setting up an Environment

- BCContainerHelper: NewBCContainer

```powershell
$startTime = [DateTime]::Now
Write-Host -ForegroundColor Green "$($startTime) : $($scriptTitle) start"

$artifactUrl = Get-BCArtifactUrl -version @version -country "us" -select Latest -type Sandbox
Write-Host -ForegroundColor Yellow ($artifactUrl)

New-BCContainer `
    -accept_eula `
    -accept_outdated `
    -alwaysPull `
    -artifactUrl $artifactUrl `
    -auth NavUserPassword `
    -containerName $containername `
    -credential $credential `
    -includeTestToolkit `
    -isolation hyperv `
    -updateHosts `
    -memoryLimit 8G


$stopTime = [DateTime]::Now
$timeSpend = ($stopTime - $startTime).TotalSeconds

Write-Host -ForegroundColor Green "$($stopTime) : $($scriptTitle) took $($timespend) seconds"
```

```
-includeTestToolkit
-includeTestLibrariesOnly
-includeTestFrameworkOnly
```

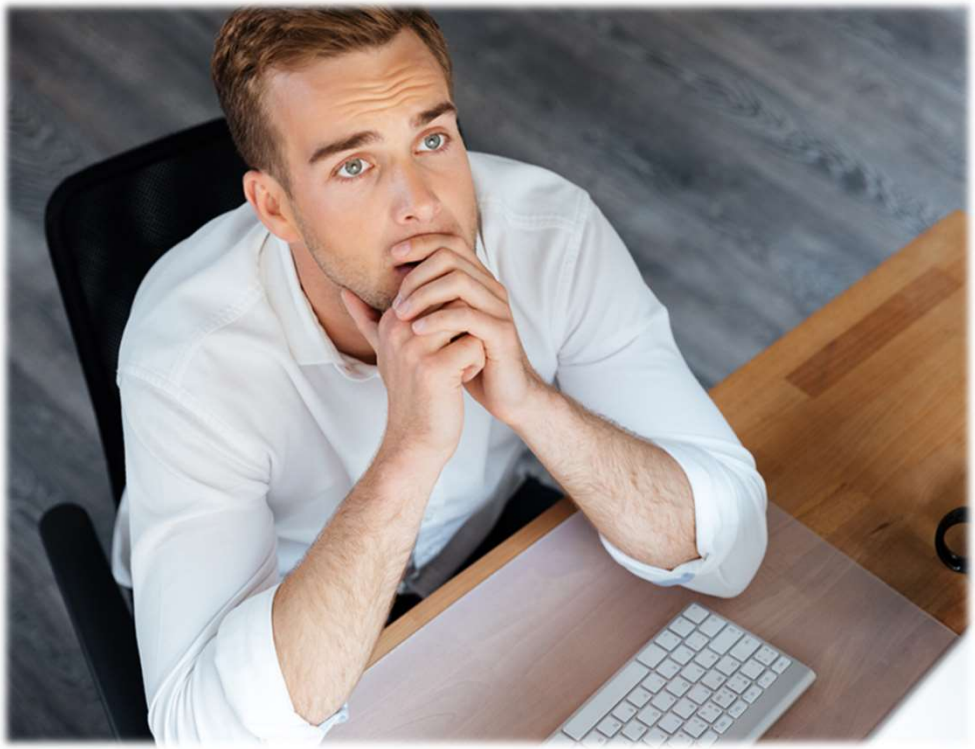
Setting up an Environment

- BCContainerHelper: ImportTestToolkitToBCContainer

```
$scriptTitle = 'Import-TestToolkitToBCContainer'

$startTime = [DateTime]::Now
Write-Host -ForegroundColor Green "$($startTime) : $($scriptTitle) start"

Import-TestToolkitToBCContainer -containerName $containerName

$stopTime = [DateTime]::Now
$timeSpend = ($stopTime -$startTime).TotalSeconds

Write-Host -ForegroundColor Green "$($stopTime) : $($scriptTitle) took $($timeSpend) seconds"
```

Writing Tests

- Where do you start?
  - Testing Plan
  - Testing Scenarios
  - Code Coverage
  - Test Driven Development



Writing Tests

- What do we test?
  - Test Pages
  - Performance Toolkit Extension
  - Test Codeunits
  - UI Handlers

Writing Tests

- TestApp Dependencies

Writing Tests

- Test Codeunits: Subtype Property

```
0 references
codeunit 80000 "TIG TestApp Event Unit"
{
    Subtype = Test;

    var
        8 references
        Assert: Codeunit Assert;
        1 reference
        LibraryERM: Codeunit "Library - ERM";
        3 references
        LibraryRandom: Codeunit "Library - Random";
        12 references
        LibrarySales: Codeunit "Library - Sales";
        15 references
        TestAppLibrary: Codeunit "TIG TestApp Library";
        7 references
        FieldValueErr: label '%1 must be %2.', Comment = '%1 Field Caption %2 Field Value';

    [Test]
    0 references
    procedure ModifyCustomer()
    var
        Customer: Record Customer;
    begin
        // [SCENARIO] A validated customer is modified
        // [GIVEN] Validation is enabled
        // [GIVEN] Validated customer exits
        TestAppLibrary.EnableValidation(true, false, false);

        LibrarySales.CreateCustomerWithAddress(Customer);

        TestAppLibrary.ValidateCustomer(Customer);

        // [WHEN] Customer is modified
        Customer.Validate(Address, LibraryRandom.RandText(MaxStrLen(Customer.Address)));

        // [THEN] Customer is not validated
        Assert.IsFalse(Customer."TIG Address Validated", StrSubstNo(FieldValueErr, Customer.FieldCaption("
    end;
```

Writing Tests

- Test Codeunits: Test Attribute

```
0 references
codeunit 80000 "TIG TestApp Event Unit"
{
    Subtype = Test;

    var
        8 references
        Assert: Codeunit Assert;
        1 reference
        LibraryERM: Codeunit "Library - ERM";
        3 references
        LibraryRandom: Codeunit "Library - Random";
        12 references
        LibrarySales: Codeunit "Library - Sales";
        15 references
        TestAppLibrary: Codeunit "TIG TestApp Library";
        7 references
        FieldValueErr: label '%1 must be %2.', Comment = '%1 Field Caption %2 Field Value';

    [Test]
    0 references
    procedure ModifyCustomer()
    var
        Customer: Record Customer;
    begin
        // [SCENARIO] A validated customer is modified
        // [GIVEN] Validation is enabled
        // [GIVEN] Validated customer exits
        TestAppLibrary.EnableValidation(true, false, false);

        LibrarySales.CreateCustomerWithAddress(Customer);

        TestAppLibrary.ValidateCustomer(Customer);

        // [WHEN] Customer is modified
        Customer.Validate(Address, LibraryRandom.RandText(MaxStrLen(Customer.Address)));

        // [THEN] Customer is not validated
        Assert.IsFalse(Customer."TIG Address Validated", StrSubstNo(FieldValueErr, Customer.FieldCaption("
    end;
```

Writing Tests

- Test Codeunits: Handler Methods

Method type	Purpose	Signature
MessageHandler	Handles Message statements.	<Function name>(<Message>: Text[1024])Error! Filename not specified.
ConfirmHandler	Handles Confirm statements.	<Function name>(<Question>: Text[1024]; var <Reply>: Boolean)
StrMenuHandler	Handles StrMenu statements.	<Function name>(<Options: Text[1024]; var <Choice>: Integer; <Instruction>: Text[1024])
ReportHandler	Handles specific reports. If you create a ReportHandler method, then that method replaces all code for running the report, including the request page, and a RequestPageHandler isn't called. Only create a RequestPageHandler if you aren't using a ReportHandler.	<Function name>(var <Report>: Report <report id>)

Source https://learn.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/devenv-creating-handler-methods

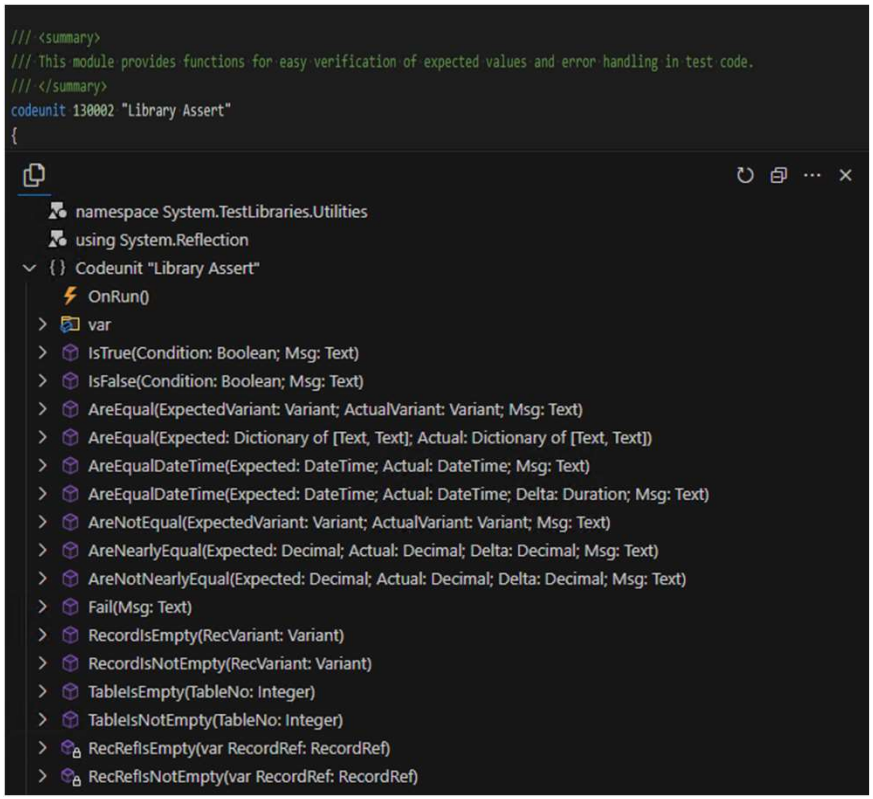
Writing Tests

- Gherkin Scenario

```
    // [SCENARIO] A description of the behavior you're going to test
    // [GIVEN] Beginning state of the scenario; what is needed to run the test
    // [WHEN] A specific action the user takes; Something Happens
    // [THEN] A testable outcome; usually caused by the [WHEN] Action; Do your test
```

Writing Tests

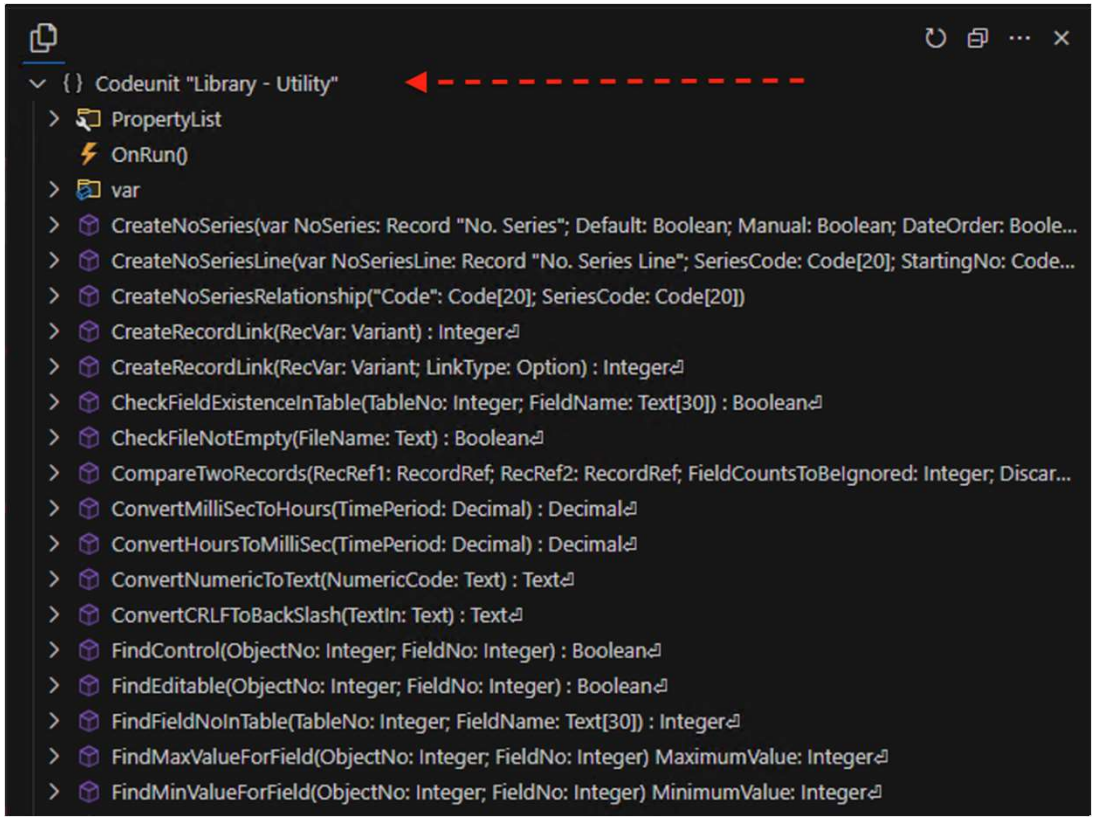
- Assert Library



Writing Tests

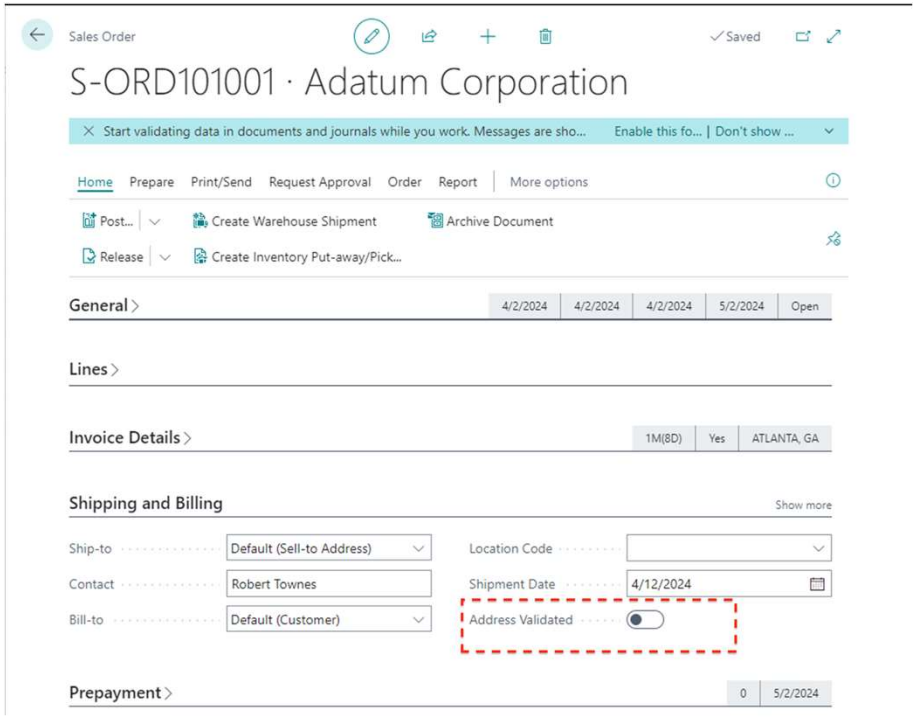
Microsoft_Tests-TestLibraries

CU	130440	Library - Random
CU	130509	Library – Sales
CU	130512	Library - Purchase
CU	131000	Library – Utility



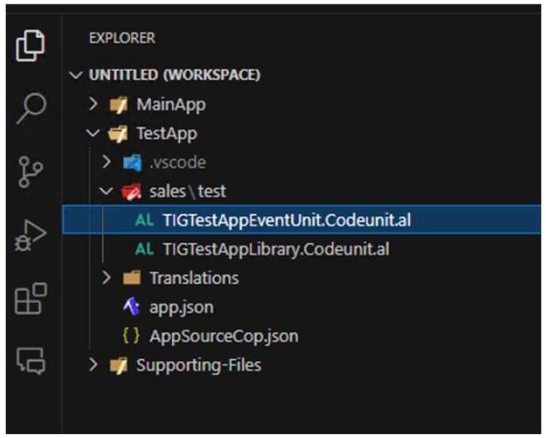
Writing Tests

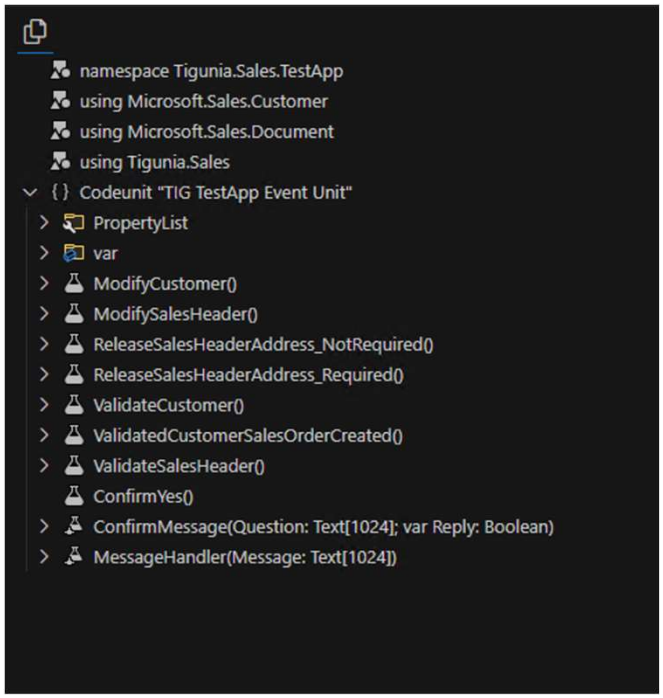
- Example App: MainApp



Writing Tests

- Example App: TestApp

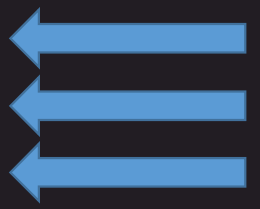




Writing Tests

- Example App: TestApp Methods

```
[Test]
0 references
procedure ModifyCustomer()
var
    Customer: Record Customer;
begin
    // [SCENARIO] A validated customer is modified
    // [GIVEN] Validation is enabled
    // [GIVEN] Validated customer exits
    TestAppLibrary.EnableValidation(true, false, false);

    LibrarySales.CreateCustomerWithAddress(Customer);

    TestAppLibrary.ValidateCustomer(Customer);

    // [WHEN] Customer is modified
    Customer.Validate(Address, LibraryRandom.RandText(MaxStrLen(Customer.Address)));

    // [THEN] Customer is not validated
    Assert.IsFalse(Customer."TIG Address Validated", StrSubstNo(FieldValueErr, Customer.FieldCaption("TIG Address Validated"), false));
end;
```



Writing Tests

- Example App: TestApp Methods

```
[Test]
0 references
procedure ModifySalesHeader()
var
    Customer: Record Customer;
    SalesHeader: Record "Sales Header";
    CustomerNo: Code[20];
begin
    // [SCENARIO] A validated sales header is modified
    // [GIVEN] Validation is enabled
    // [GIVEN] Validated sales header exists
    TestAppLibrary.EnableValidation(true, false, false);

    LibrarySales.CreateCustomerWithAddress(Customer);
    CustomerNo := Customer."No.";

    LibrarySales.CreateSalesHeader(SalesHeader, "Sales Document Type"::Order, CustomerNo);

    TestAppLibrary.ValidateSalesHeader(SalesHeader);

    // [WHEN] Sales header is modified
    SalesHeader.Validate("Ship-to Address", LibraryRandom.RandText(MaxStrLen(SalesHeader."Ship-to Address")));

    // [THEN] Sales header is not validated
    Assert.IsFalse(SalesHeader."TIG Address Validated", StrSubstNo(FieldValueErr, SalesHeader.FieldCaption("TIG Address Validated"), false));
end;
```

Writing Tests

- Example App: TestApp Methods

```
[Test]
0 references
procedure ReleaseSalesHeaderAddress_Required()
var
    Customer: Record Customer;
    SalesHeader: Record "Sales Header";
    SalesLine: Record "Sales Line";
    CustomerNo: Code[20];
    GLAccNo: Code[20];

begin
    // [SCENARIO] User releases sales order that is not validated and validation on release is enabled
    // [GIVEN] Validation is enabled and required on release
    // [GIVEN] Sales order exists and is not validated
    TestAppLibrary.EnableValidation(true, true, false);

    LibrarySales.CreateCustomerWithAddress(Customer);
    CustomerNo := Customer."No.";

    GLAccNo := LibraryERM.CreateGLAccountWithSalesSetup();

    LibrarySales.CreateSalesHeader(SalesHeader, "Sales Document Type"::Order, CustomerNo);

    LibrarySales.CreateSalesLine(SalesLine, SalesHeader, "Sales Line Type"::"G/L Account", GLAccNo, LibraryRandom.RandInt(5));

    // [WHEN] Sales Header released
    TestAppLibrary.ReleaseSalesDocumentCheckSkip(SalesHeader, false, true);

    // [THEN] Sales Header is not relesed
    Assert.IsFalse(SalesHeader.Status = "Sales Document Status"::Released, StrSubstNo(FieldValueErr, SalesHeader.FieldCaption(Status), "Sales Document Status"::Open));
end;
```

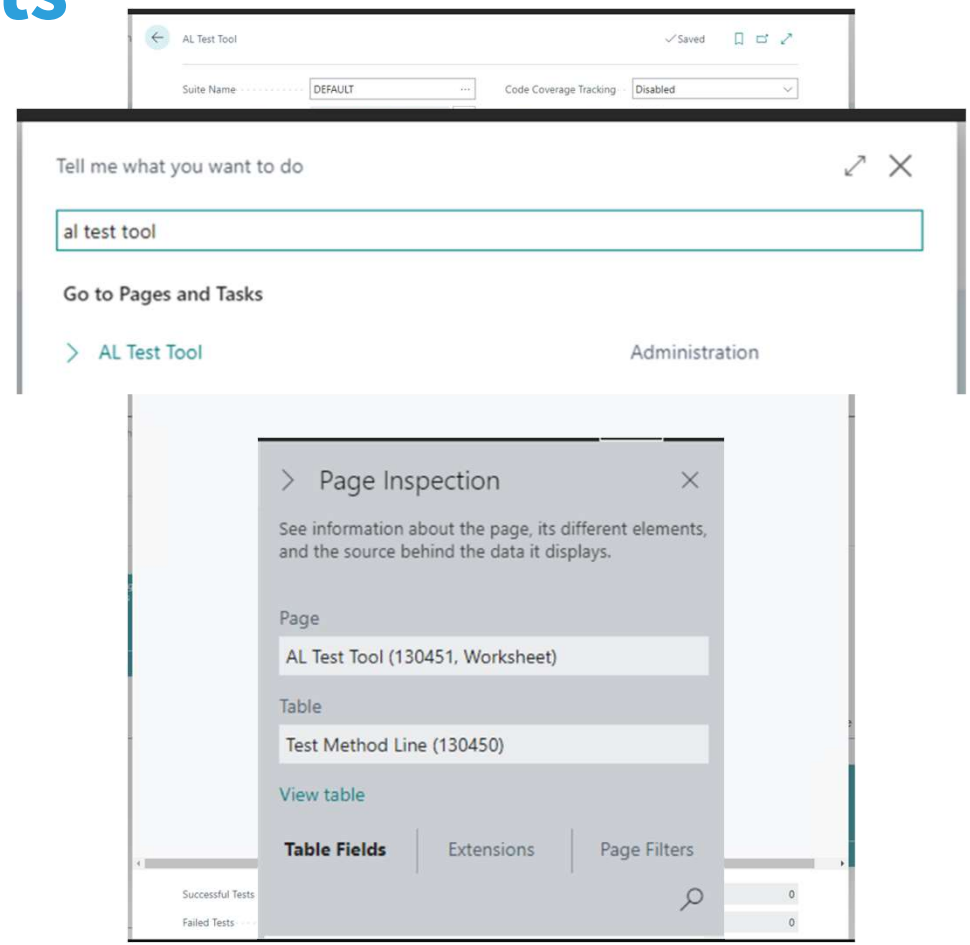
Writing Tests

- Example App: TestApp Handlers

```
[Test]
[HandlerFunctions('MessageHandler,ConfirmMessage')]
0 references
procedure ConfirmYes()
begin
    // [SCENARIO] Demonstrate handler methonds
    // [GIVEN] Handlers assigned
    // [WHEN] Test confirm and messaege
    TestAppLibrary.AskConfirmMessage();
    // [THEN] No Assert
end;

[ConfirmHandler]
0 references
procedure ConfirmMessage(Question: Text[1024]; var Reply: Boolean);
var
    SalesManagement: Codeunit "TIG Sales Management";
begin
    Reply := (Question = SalesManagement.GetConfirmMsg());
end;

[MessageHandler]
0 references
procedure MessageHandler(Message: Text[1024])
var
    SalesManagement: Codeunit "TIG Sales Management";
begin
    Assert.IsTrue(StrPos(Message, SalesManagement.GetYesMsg()) > 0, Message);
end;
```

Running Tests

- AL Test Tool



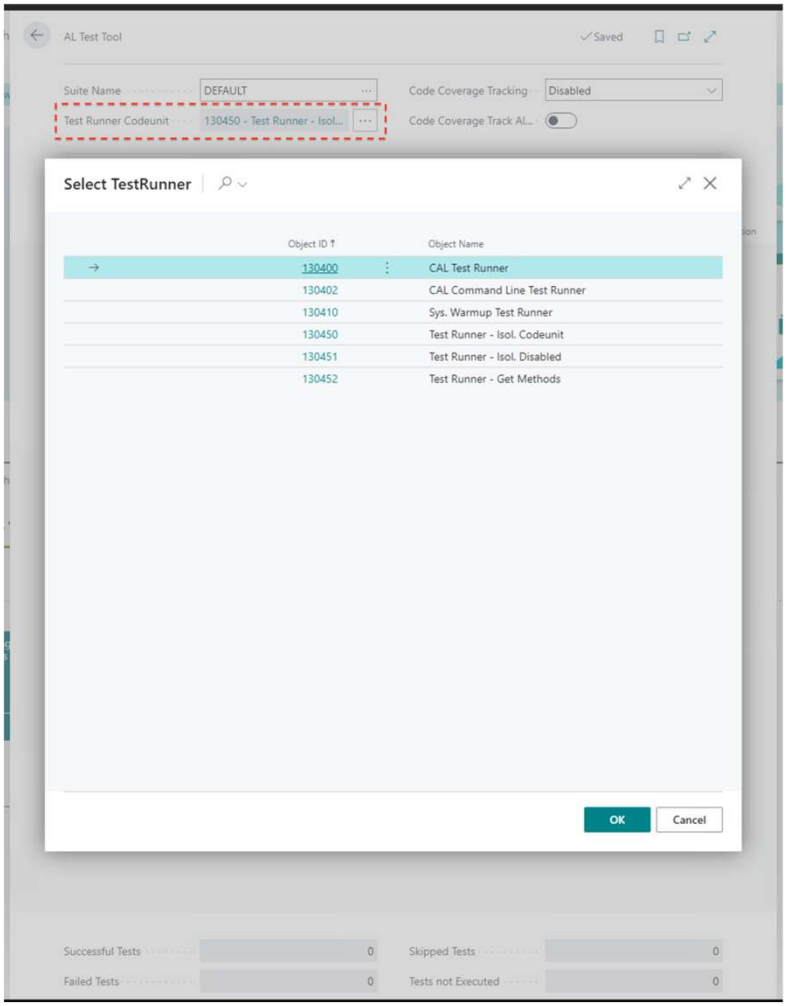
Running Tests

- Test Runner Codeunit

TestIsolation Property

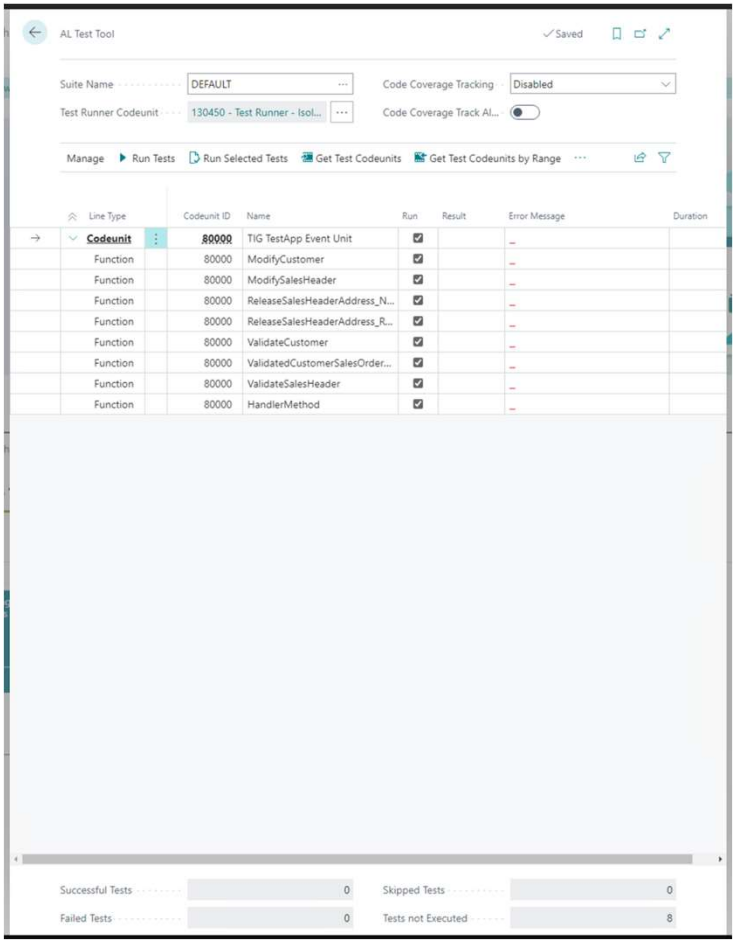
Value	Description
Disabled	Do not roll back any changes to the database. Tests are not isolated from each other. This is the default value.
Codeunit	Roll back all changes to the database after each test codeunit executes.
Function	Roll back all changes to the database after each test method executes.

Source https://learn.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/properties/devenv-testisolation-property



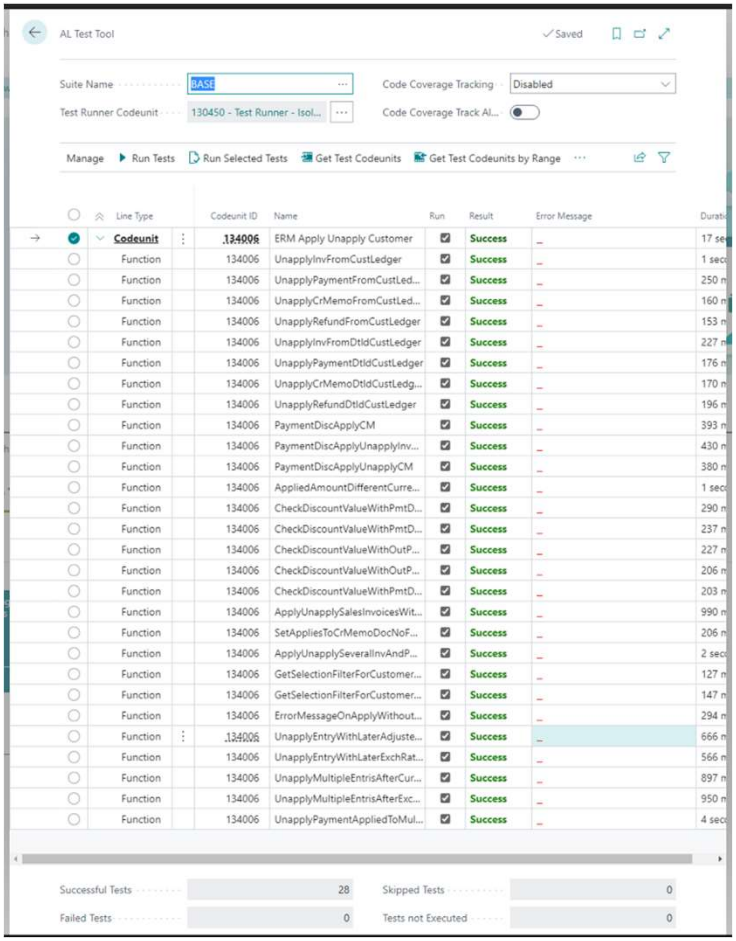
Running Tests

- Test Codeunits



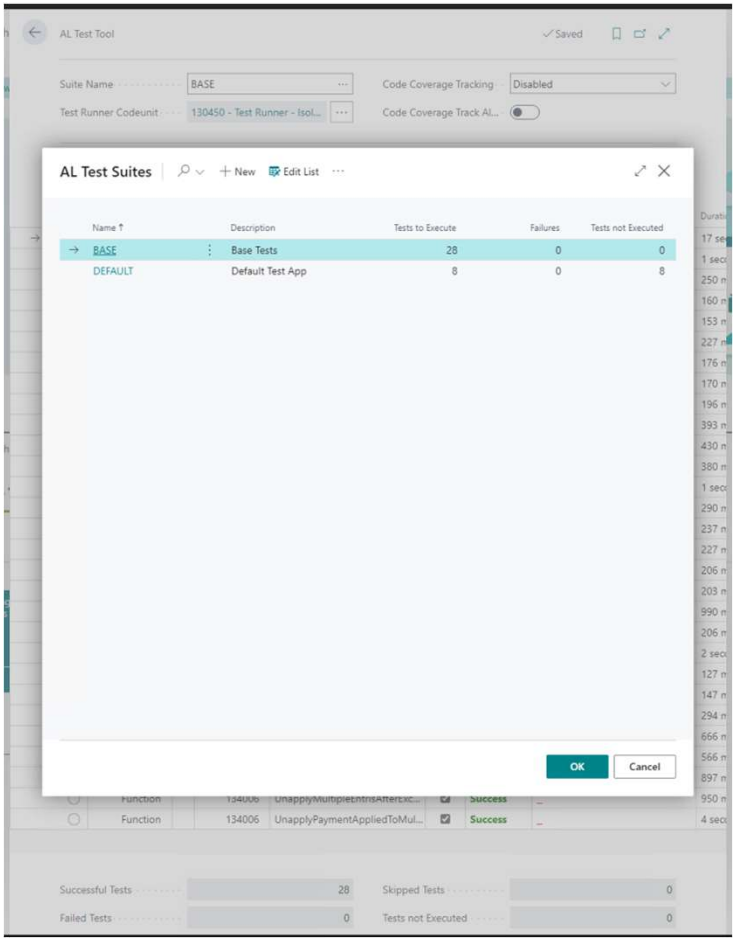
Running Tests

- Base Tests



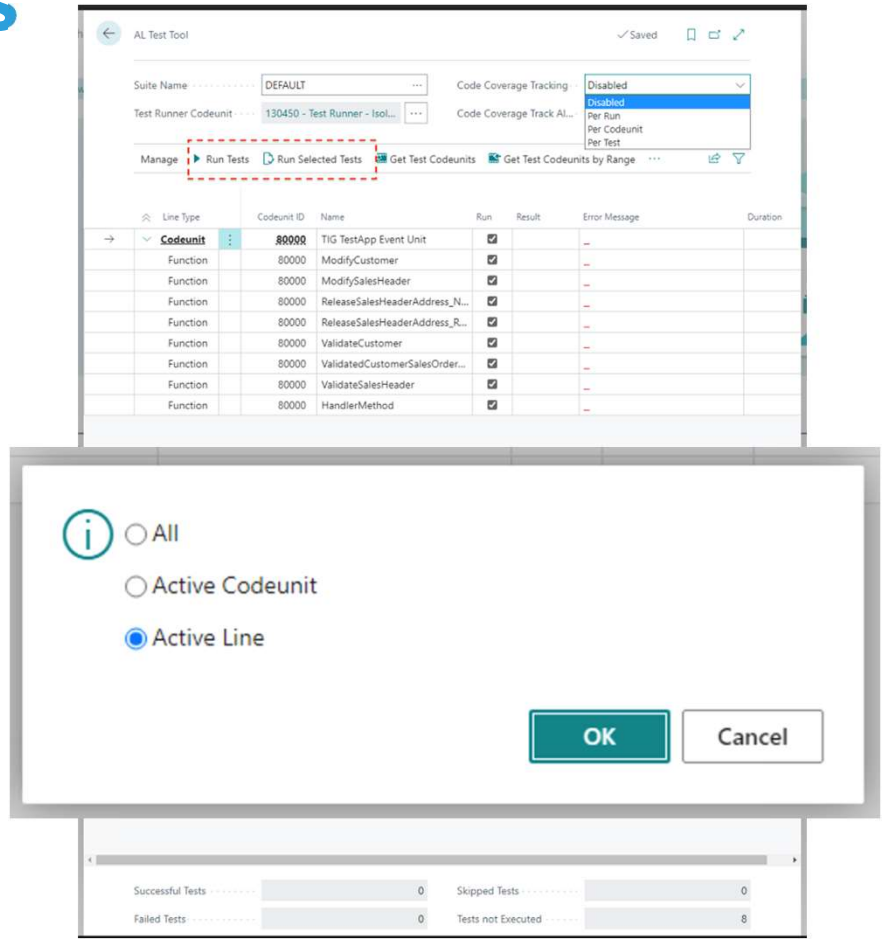
Running Tests

- Test Suite



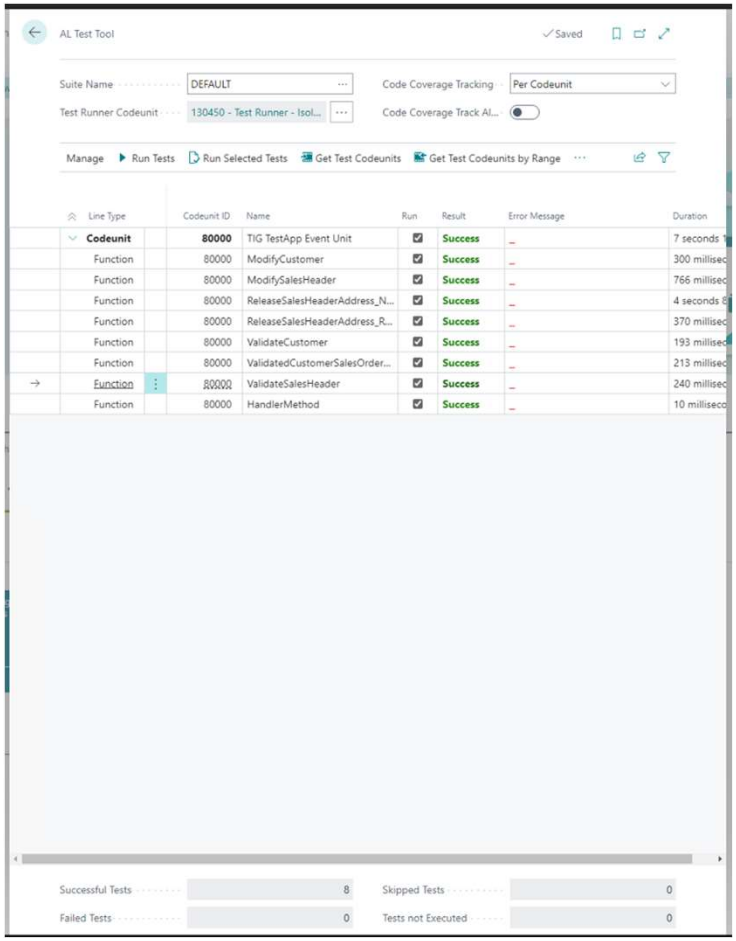
Running Tests

- Running Tests



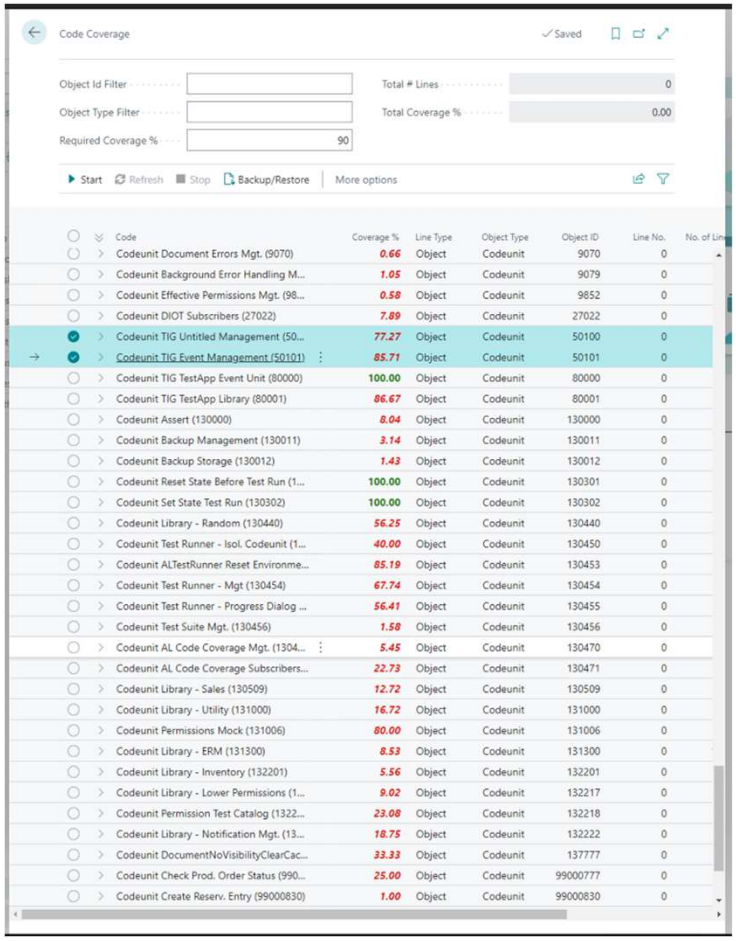
Running Tests

- Completed Tests



Running Tests

- Code Coverage



Running Tests

- BCContainerHelper: Get-TestsFromBCContainer

```
Get-TestsFromBCContainer `
    -containerName $containerName `
    -Credential $credential `
    -testSuite $testSuite `
    -ignoreGroups
```

```
Tests                                                
```

Running Tests

- BCContainerHelper: Get-TestsFromBCContainer

```
$tests = Get-TestsFromBCContainer `
    -containerName $containerName `
    -Credential $credential `
    -testSuite $testSuite `
    -ignoreGroups

foreach( $test in $tests) {
    Write-Host -ForegroundColor Yellow $tests.Id $tests.Name
    Write-Output $test.Tests
```

```
80000 TIG TestApp Event Unit
ModifyCustomer
ModifySalesHeader
ReleaseSalesHeaderAddress_NotRequired
ReleaseSalesHeaderAddress_Required
ValidateCustomer
ValidatedCustomerSalesOrderCreated
ValidateSalesHeader
ConfirmYes
```

Running Tests

- BCContainerHelper: Run-TestsInBCContainer

```
Run-TestsInBCContainer `
    -containerName $containerName `
    -credential $credential `
    -testSuite $testSuite `
    -detailed `
    -AppendToXUnitResultFile:($true) `
    -XUnitResultFileName $xunitResultsFile
```

```
Codeunit 80000 TIG TestApp Event Unit Success (3.875 seconds)
  Testfunction ModifyCustomer Success (0.66 seconds)
  Testfunction ModifySalesHeader Success (0.233 seconds)
  Testfunction ReleaseSalesHeaderAddress_NotRequired Success (0.4 seconds)
  Testfunction ReleaseSalesHeaderAddress_Required Success (0.244 seconds)
  Testfunction ValidateCustomer Success (0.18 seconds)
  Testfunction ValidatedCustomerSalesOrderCreated Success (0.163 seconds)
  Testfunction ValidateSalesHeader Success (0.213 seconds)
  Testfunction ConfirmYes Success (0.007 seconds)
```

Running Tests

- BCContainerHelper: Run-TestsInBCContainer

```powershell
$first = $true
$tests = Get-TestsFromBCContainer `
    -containerName $containerName `
    -Credential $Credential `
    -testSuite $testSuite `
    -ignoreGroups

$tests | ForEach-Object {
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<assemblies>
  <assembly name="80000 TIG TestApp Event Unit" test-framework="PS Test Runner" run-date="2023-04-10" run-time="17:49:08" total="8" passed="8" failed="0" skipped="0" time="7.777">
    <collection name="TIG TestApp Event Unit" total="8" passed="8" failed="0" skipped="0" time="7.777" Skipped="0">
      <test name="TIG TestApp Event Unit:ModifyCustomer" method="ModifyCustomer" time="2.627" result="Pass" />
      <test name="TIG TestApp Event Unit:ModifySalesHeader" method="ModifySalesHeader" time="1.983" result="Pass" />
      <test name="TIG TestApp Event Unit:ReleaseSalesHeaderAddress_NotRequired" method="ReleaseSalesHeaderAddress_NotRequired" time="2.01" result="Pass" />
      <test name="TIG TestApp Event Unit:ReleaseSalesHeaderAddress_Required" method="ReleaseSalesHeaderAddress_Required" time="0.4" result="Pass" />
      <test name="TIG TestApp Event Unit:ValidateCustomer" method="ValidateCustomer" time="0.256" result="Pass" />
      <test name="TIG TestApp Event Unit:ValidatedCustomerSalesOrderCreated" method="ValidatedCustomerSalesOrderCreated" time="0.24" result="Pass" />
      <test name="TIG TestApp Event Unit:ValidateSalesHeader" method="ValidateSalesHeader" time="0.247" result="Pass" />
      <test name="TIG TestApp Event Unit:HandlerMethod" method="HandlerMethod" time="0.014" result="Pass" />
    </collection>
  </assembly>
</assemblies>
```

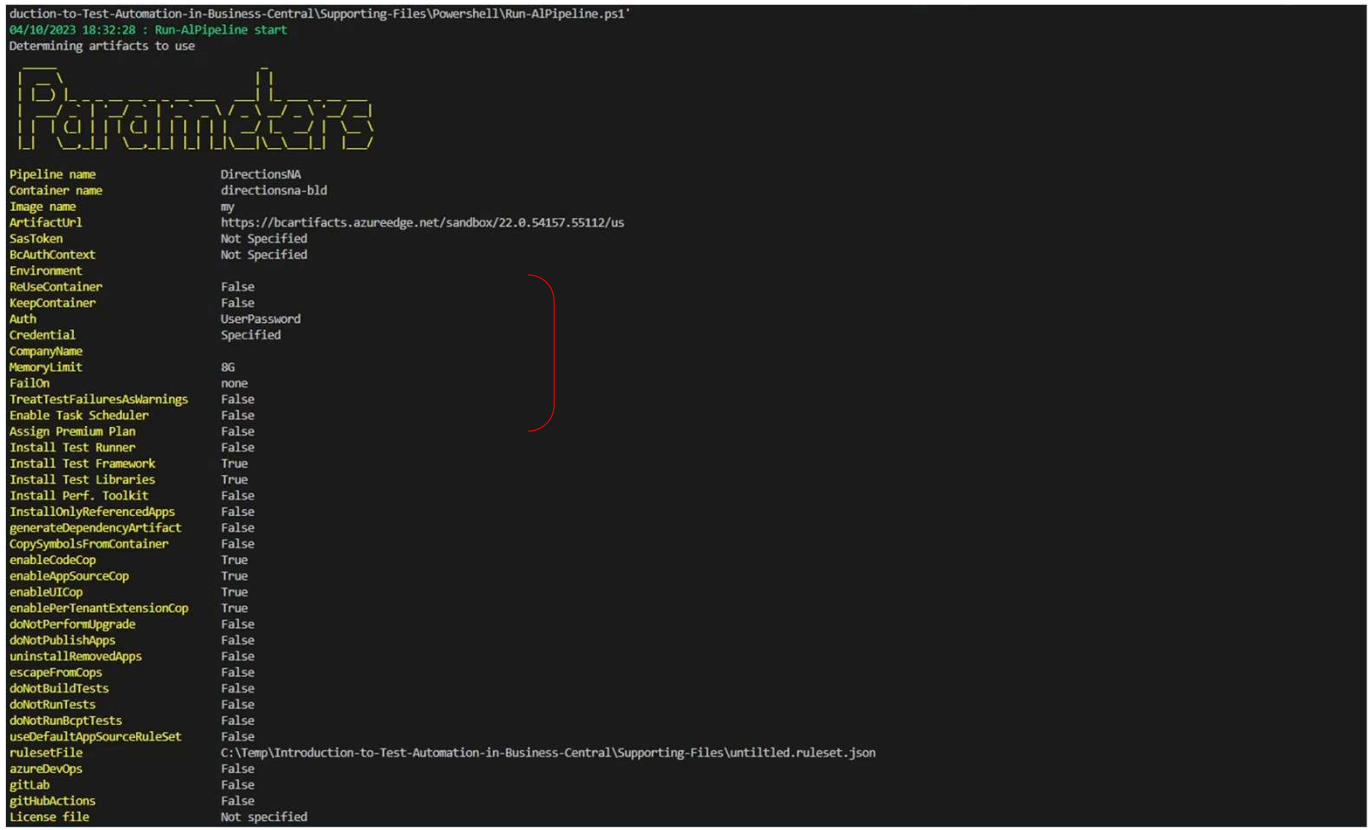
Running Tests

- BCContainerHelper: Run-AlPipeline

```
##########################################################################################

$scriptTitle = 'Run-AlPipeline'

$startTime = [DateTime]::Now
Write-Host -ForegroundColor Green "$($startTime) : $($scriptTitle) start"

Run-AlPipeline `
    -pipelineName $pipelineName `
    -baseFolder $baseFolder `
    -appFolders $appFolders `
    -testFolders $testFolders `
    -installTestFramework `
    -installTestLibraries `
    -enablePerTenantExtensionCop `
    -enableUICop `
    -enableCodeCop `
    -enableAppSourceCop `
    -appVersion $appVersion `
    -Credential $credential `
    -CreateRuntimePackages `
    -AppSourceCopMandatoryAffixes $affix `
    -AppSourceCopSupportedCountries "US" `
    -rulesetFile $rulesetFile `
    -outputFolder $outputFolder `
    -testResultsFile $testResultsFile `
    -packagesFolder $PackagesFolder


$stopTime = [DateTime]::Now
$timeSpend = ($stopTime -$startTime).TotalSeconds

Write-Host -ForegroundColor Green "$($stopTime) : $($scriptTitle) took $($timeSpend) seconds"
```

Running Tests

- BCContainerHelper: Run-AlPipeline

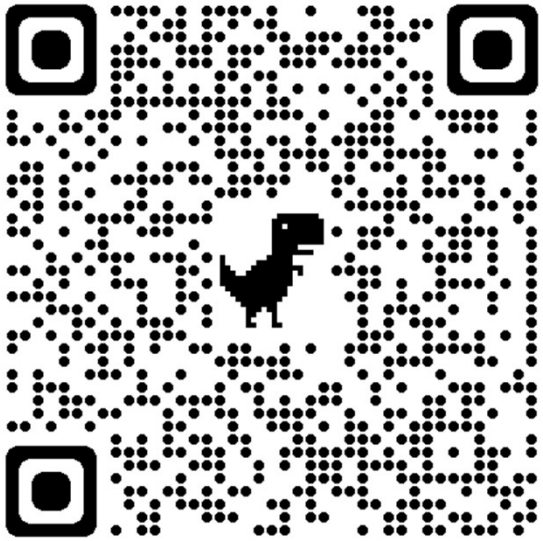


Questions?

What's Next

- Establish Test Automation Policy
- Educate
- Explore

References



Brad Prendergast

Thank You!

Please remember to fill out your session survey using the Directions 2024 Mobile App!