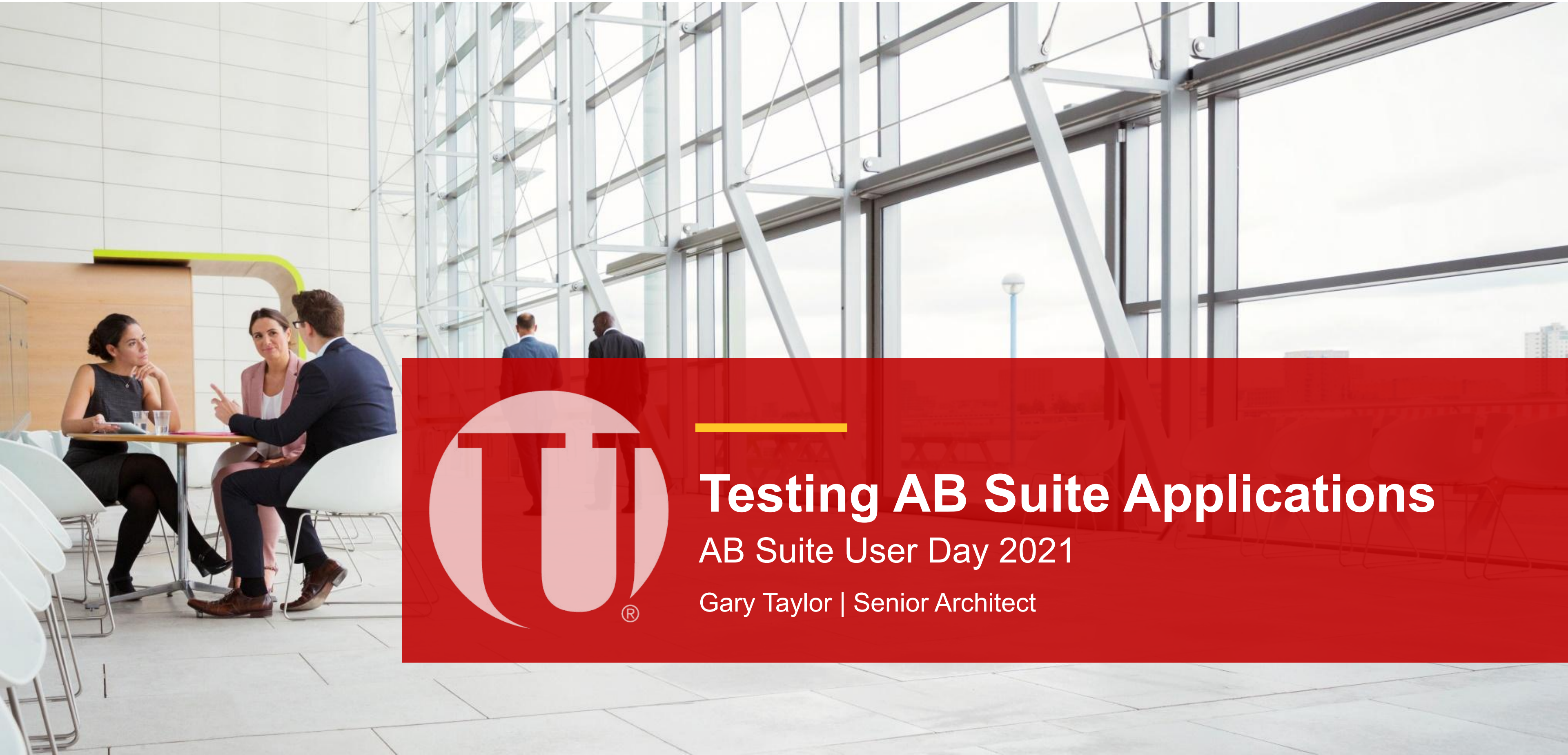
Securing Your Tomorrow®

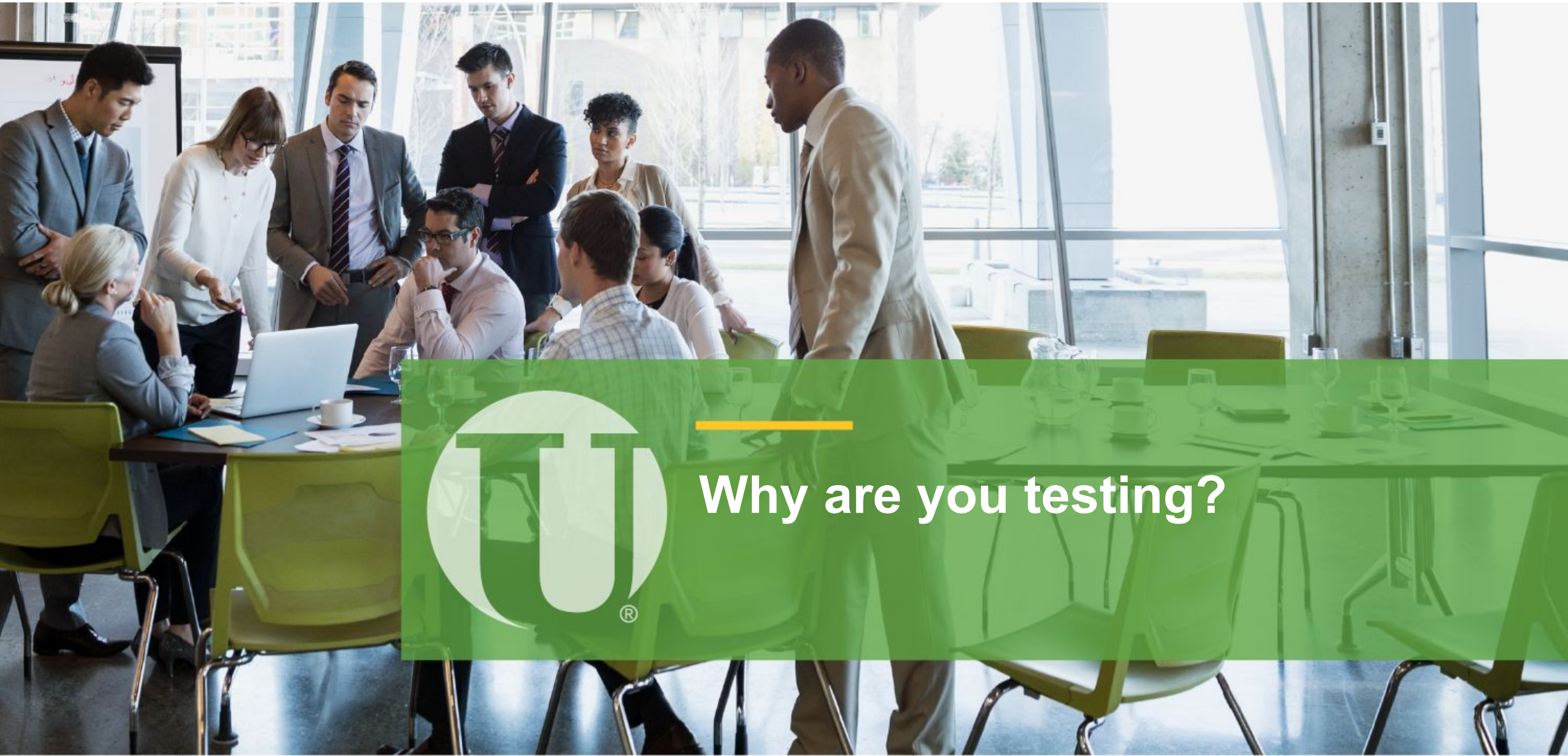


Testing AB Suite Applications

AB Suite User Day 2021

Gary Taylor | Senior Architect

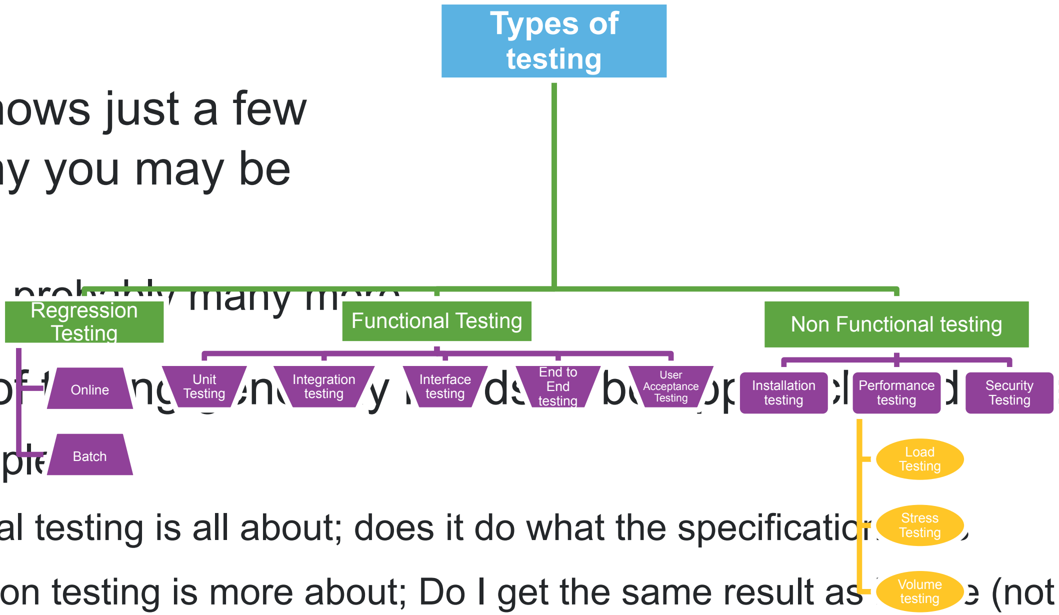
UNISYS | Securing Your Tomorrow®



Why are you testing?

Testing for many reasons

- Diagram shows just a few reasons why you may be testing
  - There are probably many more
- Each type of t[illegible]ng [illegible]en[illegible]y [illegible]ds [illegible] b[illegible]p[illegible]c[illegible]d[illegible]ently
  - For Example
    - Functional testing is all about; does it do what the specificatio[illegible]
    - Regression testing is more about; Do I get the same result as [illegible]e (not necessarily correct result!)
  - Automation may not be reusable across different type of testing
    - Unlikely to be able to reuse Unit tests easily to Volume test for example



Reality – no one size fits all; need to consider best approach for each type of testing

Testing of the Batch

- Relatively easy to automate the running of Batch
  - You probably already do it in production
- But comparing results generally difficult
  - Results usually a mixture of
    - Print files
    - Text/Output files
    - Database updates
  - Very few tools specifically aimed at helping with this
    - So difficult to automate

Testing Online applications

- By contrast Online testing tends to be more challenging
  - Multiple different interfaces to consider
    - End to End versus Back End
    - Character mode
    - Graphical
    - Web
    - B2B
- But generally easier to compare the results
  - For inputs A,B,C I got back values X,Y,Z is that as expected or same as last time?
- Lots of tools that suggest they can help automate
  - Not all ideal with AB Suite or all types of interface
  - Challenge creating or preparing the data to automate
    - So often still done manually

Ideas on how to improve your testing

What's in the Box

Think outside the Box

ATT

Further Options

UNISYS | Securing Your Tomorrow®

What’s in the Box

Debugger

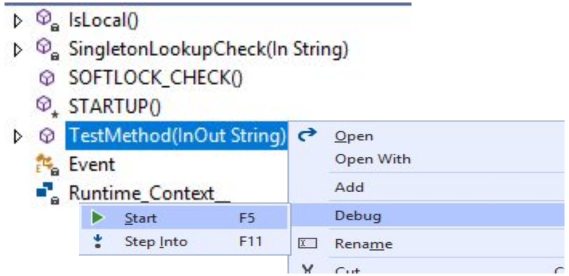
- Primarily for Unit Testing
  - Test individual objects
    - Reports
    - Methods
    - System
- Local (SQL Server Database) or connect to runtime database on the host server
  - Easy to populate via standard tools
- Offers RATL interface
  - So can be used for testing interfaces as well

How to Debug a Method()

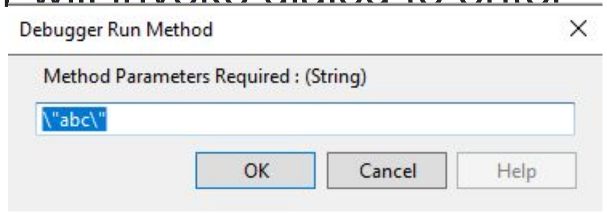
- AB Suite allows you to Debug **Public** Segment level Methods
  - Can pass in Primitive parameters

- No need to invoke Framework cycle to test

- To use - Right Click => Debug



- Any Parameter will invoke dialog to enter

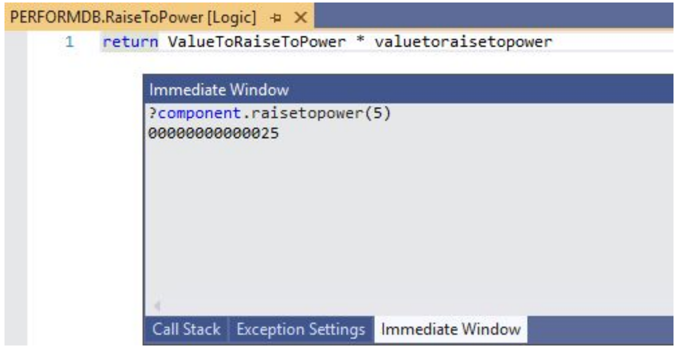

Debugger Run Method

Method Parameters Required : (String)

\"abc\"

Other ways to Debug a Non Public Segment Method

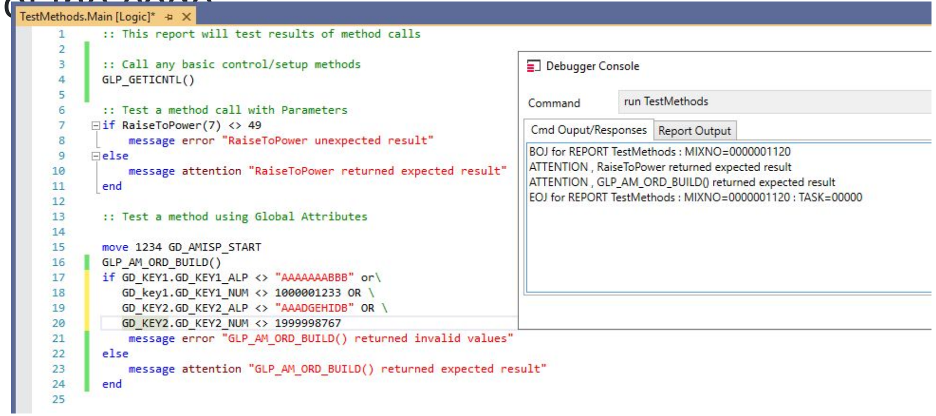
- Use Immediate Window in Debugger to return result of Method call
  - Can’t step through logic but ideal for verifying results



```
return ValueToRaiseToPower * valuetoraisetopower
```

```
?component.raisetopower(5)
00000000000025
```

- Create Test Harness Report that calls the Methods
  - Validate the results as part of the code
  - Make adding new methods to Test harness a development standards



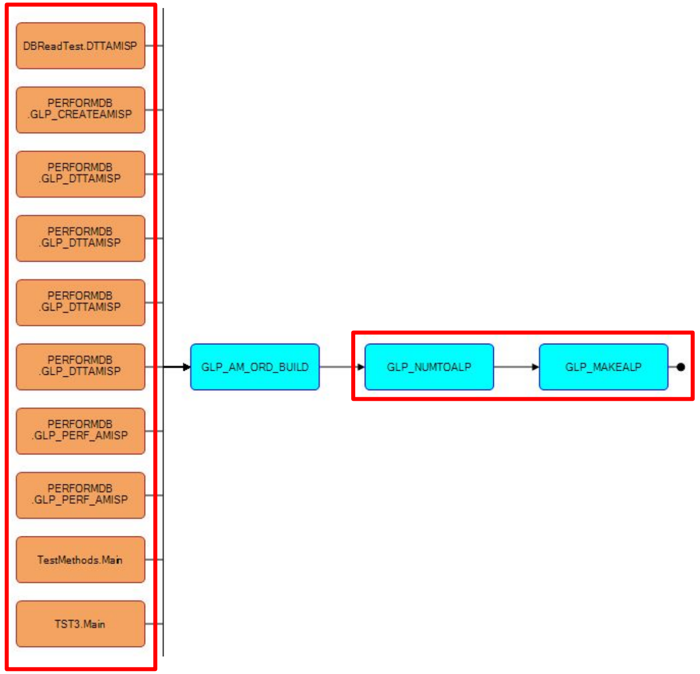
```
:: This report will test results of method calls

:: Call any basic control/setup methods
GLP_GETICNTL()

:: Test a method call with Parameters
if RaiseToPower(7) <> 49
    message error "RaiseToPower unexpected result"
else
    message attention "RaiseToPower returned expected result"
end

:: Test a method using Global Attributes

move 1234 GD_AMISP_START
GLP_AM_ORD_BUILD()
if GD_KEY1.GD_KEY1_ALP <> "AAAAAAABBB" or\
   GD_key1.GD_KEY1_NUM <> 1000001233 OR \
   GD_KEY2.GD_KEY2_ALP <> "AAADGEHIDB" OR \
   GD_KEY2.GD_KEY2_NUM <> 1999998767
    message error "GLP_AM_ORD_BUILD() returned invalid values"
else
    message attention "GLP_AM_ORD_BUILD() returned expected result"
end
```

Command: run TestMethods

```
BOJ for REPORT TestMethods : MIXNO=0000001120
ATTENTION , RaiseToPower returned expected result
ATTENTION , GLP_AM_ORD_BUILD() returned expected result
EOJ for REPORT TestMethods : MIXNO=0000001120 : TASK=00000
```

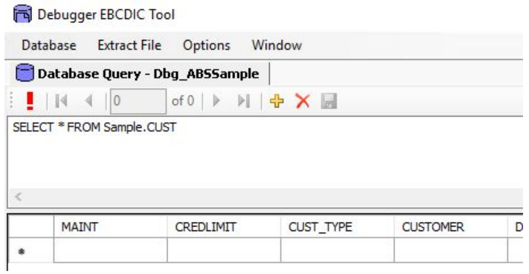
So why test Methods?

- The example on previous slide tested Method **GLP_AM_ORD_BUILD()**
- We see from the MATRIX Method Call Graph
  - Method calls two other methods
  - Method Called by multiple other Methods
- So testing of methods
  - Wide test coverage
  - Often simple to implement
  - Another approach to testing that is easy to expand



Choose what platform you emulate

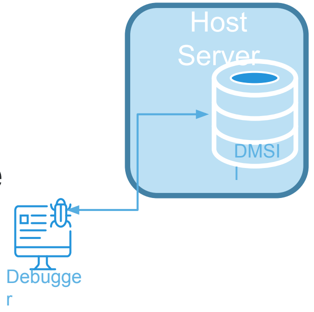
- Solution platform determines how data stored by Debugger
  - Platform MCP - data (DB and Extract files) in EBCDIC
    - Data returned in same collating sequence as on MCP host
    - EBCDIC Tool supplied to help view and manipulate data
  - For other platforms data stored in ASCII
    - Use standard Windows tools to manipulate data e.g. SQL Server Management Studio (SSMS)



```
SELECT * FROM Sample.CUST
```

	MAINT	CREDLIMIT	CUST_TYPE	CUSTOMER	D
*					

- Debug against the Host database – HDBA
  - For MCP uses DMSII OLEDB driver
  - Windows point Debugger at Target Runtime database
  - Coming for OS 2200



Populating Local Debugger Databases is easy!

- **UK Developed tool** to help transfer data between different source/targets
  - Auto Generates SSIS Packages to move data
  - Understands different platform Schema definitions
  - Any co on allowed

Source Data Type	Target Destinations
SQL Server (EAE or AB Suite)	SQL Server (AB Suite) Optional EBCDIC
MCP (EAE/AB Suite)	MCP (EAE/AB Suite)
Oracle (EAE)	EAE Dev Test
OS 2200	Flat Files (CSV)
EAE Developer Test	

  - Capability to Filter data on transfer
    - Use simple rules to determine what rows are sent to target database
    - E.g. *SUBSTR(SUB_ACC_NO,1,5) = "A1234"*

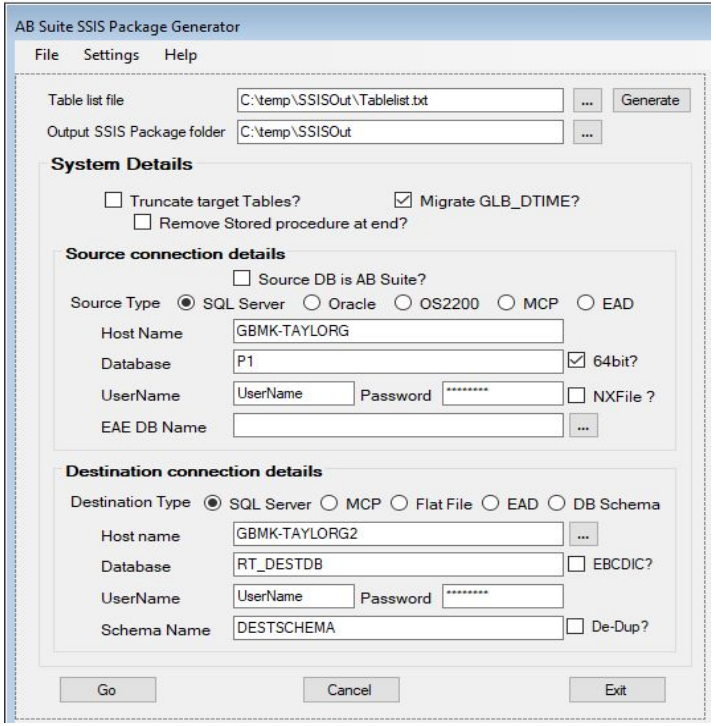


Table list file: C:\temp\SSISOut\Tablelist.txt
Output SSIS Package folder: C:\temp\SSISOut

System Details

☐ Truncate target Tables? ☑ Migrate GLB_DTIME?
☐ Remove Stored procedure at end?

Source connection details

☐ Source DB is AB Suite?
Source Type: [✓ SQL Server] Oracle OS2200 MCP EAD
Host Name: GBMK-TAYLORG
Database: P1 ☑ 64bit?
UserName: UserName Password: ******** ☐ NXFile ?
EAE DB Name:

Destination connection details

Destination Type: [✓ SQL Server] MCP Flat File EAD DB Schema
Host name: GBMK-TAYLORG2
Database: RT_DESTDB ☐ EBCDIC?
UserName: UserName Password: ********
Schema Name: DESTSCHEMA ☐ De-Dup?

» Read More (July 2019 Developing Agility article)

Using Client Tools with Debugger

Test your Component Enabler Clients via Debugger

- Increasingly we see our Clients moving away from terminal emulator interfaces
  - Moving to GUI or B2B type interfaces i.e. HTML or Web Services
    - Via use of Component Enabler and RATL
  - Requirement to test these interfaces earlier in development cycle
- Debugger provides the ability for these client application to connect via RATL
  - Allows testing of Client application in parallel to Development of AB Suite Application
- Next few slides walk you through process to set up

Configure Adaptor (RATL)

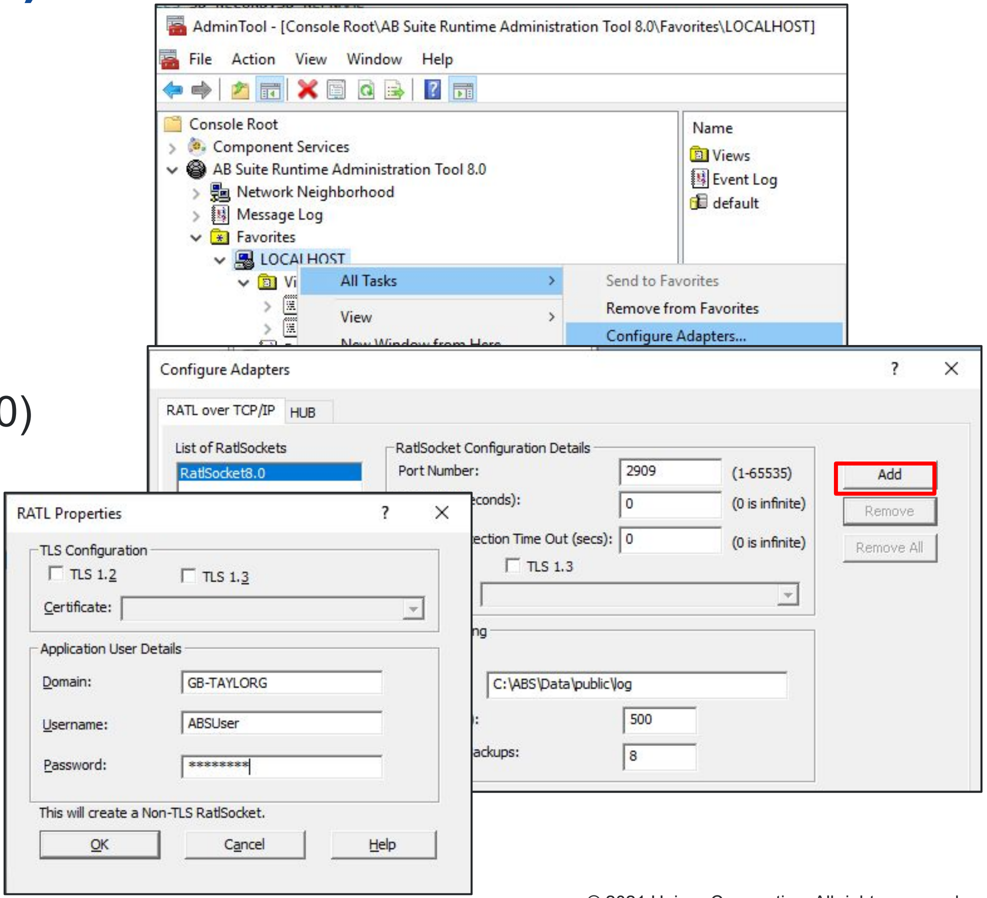
- Configure a new Adaptor via **Admin Tool**
  - Add another RATL via ADD button
  - Enter Properties
  - Configure Port (default will be 2910)
  - Creates new Windows Service
- Via Admin CMD Prompt
  - Stop new service

```
NET STOP RatlSocket8.0-1
```

  - Configure to manual start

```
RatlSocket8.0-1 is name from List of RatlSockets
```

T=DEMAND



Port Number: 2909 (1-65535)
...econds): 0 (0 is infinite)
...ection Time Out (secs): 0 (0 is infinite)
TLS 1.3
C:\ABS\Data\public\log
500
...ackups: 8

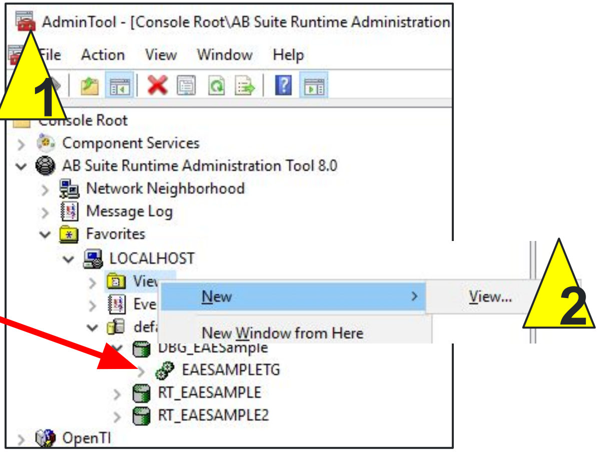
Domain: GB-TAYLORG
Username: ABSUser
Password: ********

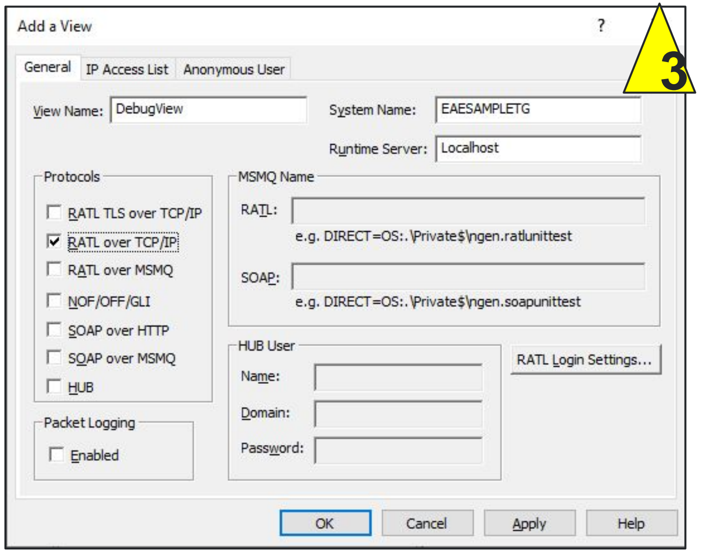
This will create a Non-TLS RatlSocket.

|

Create View

1. Start Debugger at least once
   - This will add system to Admin Tool





View Name: DebugView
System Name: EAESAMPLETG
Runtime Server: Localhost

Protocols:
- [ ] RATL TLS over TCP/IP
- [x] RATL over TCP/IP
- [ ] RATL over MSMQ
- [ ] NOF/OFF/GLI
- [ ] SOAP over HTTP
- [ ] SOAP over MSMQ
- [ ] HUB

MSMQ Name:
RATL:
e.g. DIRECT=OS:.\Private$\ngen.ratlunittest
SOAP:
e.g. DIRECT=OS:.\Private$\ngen.soapunittest

HUB User:
Name:
Domain:
Password:

Packet Logging:
- [ ] Enabled

2. Need to add a VIEW to point to Debugger system
   - Right click on Views => New View

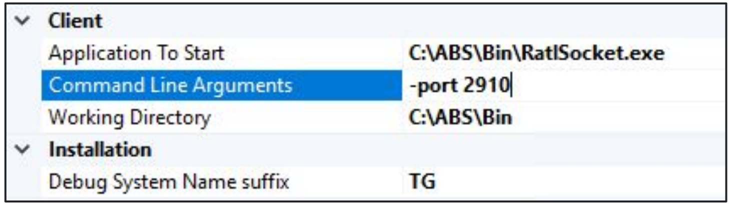
3. Define View
   - View Name (Case sensitive)
   - System/Host names
   - Protocols – RATL over TCP/IP

UNISYS | Securing Your Tomorrow®

Configure Project Configuration settings

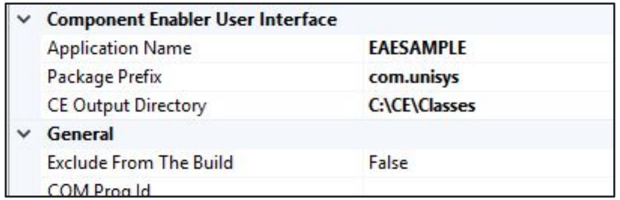
- Configure **Model** to start the RatlSocket.exe process
  - Specify the Port you want RATL to listen on

    Note …
    In releases prior to 8.0 you have to copy RatlSocket as a different name first and then run the copy



Client	
Application To Start	C:\ABS\Bin\RatlSocket.exe
Command Line Arguments	-port 2910
Working Directory	C:\ABS\Bin
Installation	
Debug System Name suffix	TG

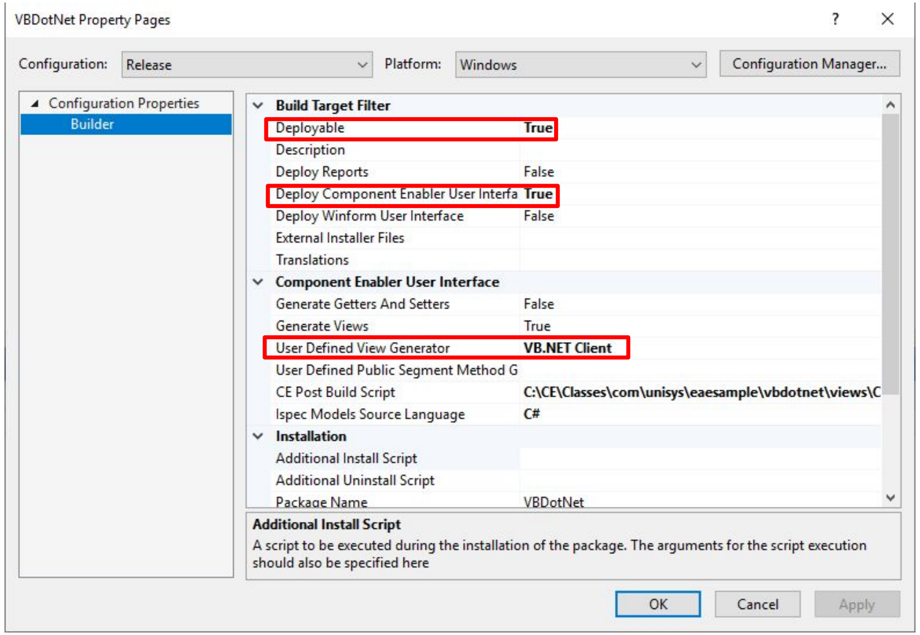
- Configure **Segment** to define CE details:
  - Application name
  - Package Prefix
  - CE Output Directory



Component Enabler User Interface	
Application Name	EAESAMPLE
Package Prefix	com.unisys
CE Output Directory	C:\CE\Classes
General	
Exclude From The Build	False
COM Prog Id	

Configure and Build Bundle

- Right click on Bundle Folder and **Build**
  - Folder will need to be set as Deployable for CE User Interface
    - Ideally configured to run post build compile scripts
  - Include all the Screens to be used
- Use **Re-Build** to force it to build all files again
- Much simpler with 7.0 onwards


VBDotNet Property Pages

Configuration: Release   Platform: Windows   Configuration Manager...

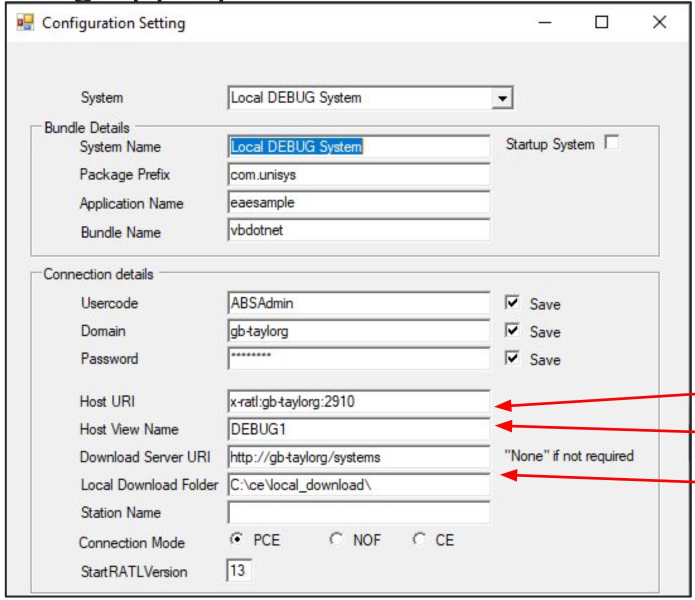
Configuration Properties
- Builder

Build Target Filter	
Deployable	**True**
Description	
Deploy Reports	False
Deploy Component Enabler User Interfa	**True**
Deploy Winform User Interface	False
External Installer Files	
Translations	
Component Enabler User Interface	
Generate Getters And Setters	False
Generate Views	True
User Defined View Generator	**VB.NET Client**
User Defined Public Segment Method G	
CE Post Build Script	C:\CE\Classes\com\unisys\eaesample\vbdotnet\views\C
Ispec Models Source Language	C#
Installation	
Additional Install Script	
Additional Uninstall Script	
Package Name	VBDotNet

Additional Install Script
A script to be executed during the installation of the package. The arguments for the script execution should also be specified here

Configure CE Client and connect

- Starting Debugger will now start RATL listener
  - Once RATL starts then connect to system via your CE Client using appropriate connection details



Configuration Setting

System	Local DEBUG System

Bundle Details

System Name	Local DEBUG System	Startup System ☐
Package Prefix	com.unisys	
Application Name	eaesample	
Bundle Name	vbdotnet	

Connection details

Usercode	ABSAdmin	☑ Save
Domain	gb-taylorg	☑ Save
Password	********	☑ Save
Host URI	x-ratl:gb-taylorg:2910	
Host View Name	DEBUG1	
Download Server URI	http://gb-taylorg/systems	"None" if not required
Local Download Folder	C:\ce\local_download\	
Station Name		
Connection Mode	[✓ PCE] NOF CE	
StartRATLVersion	13	

Define URI with RATL Port

Specify correct VIEW name

Location of generated files

Think outside the Box

Still using Terminal emulators?

- There are many 3rd party tools designed to help facilitate testing
  - Shown are just a few of the more common ones
  - The common thread for these tools is they need a Web Service or HTML (ASP) based interface
- Don’t assume you can’t use any of these if you only have Fixed mode screens.
  - AB Suite ships with standard generators to create simple ASP.Net interface or Web Services
  - Don’t need to have created GUI screens
  - Whilst communication path is different the end point is the same Ispec Logic

How to Test via ASP.Net / RATL

- Basic Steps to enable testing via ASP.Net Client and RATL

  One time steps

  1. Configure View on Host
  2. Prep Target ASP.Net Generator – using “<CE Folder>\ASP.NET Generator\Utilities\Setup\SetupASPNet.vbs“
     - Configure Web.Config file
  3. Create IIS Application
  4. Define Segment Level CE Setting – directing output to folder 2 above
  5. Add a Deployable CE Folder for ASP.Net Generator
  6. Add the Ispecs to folder that you want to test

  Usage

  1. Build System/Bundle
  2. Compile ASP.Net application

Not sure or confident about doing this then let us help

What About Web Services?

- AB Suite Client Tools includes a standard SOAP Webservices Generator
  - Would allow testing via tools like PostMan, ReadyAPI etc.
    - Ispecs have to be Stateless i.e. cant use GLB.Work to pass data

- Another possible option
  - Forthcoming Developing Agility article about “Test Gateway” **prototype**
    - Exposes a simple Microsoft WebAPI based JSON interface via a RESTFul Web Service
      - Stateful Ispec support

This is only a **Prototype** currently but if it might be of interest let us know

What about the Batch?

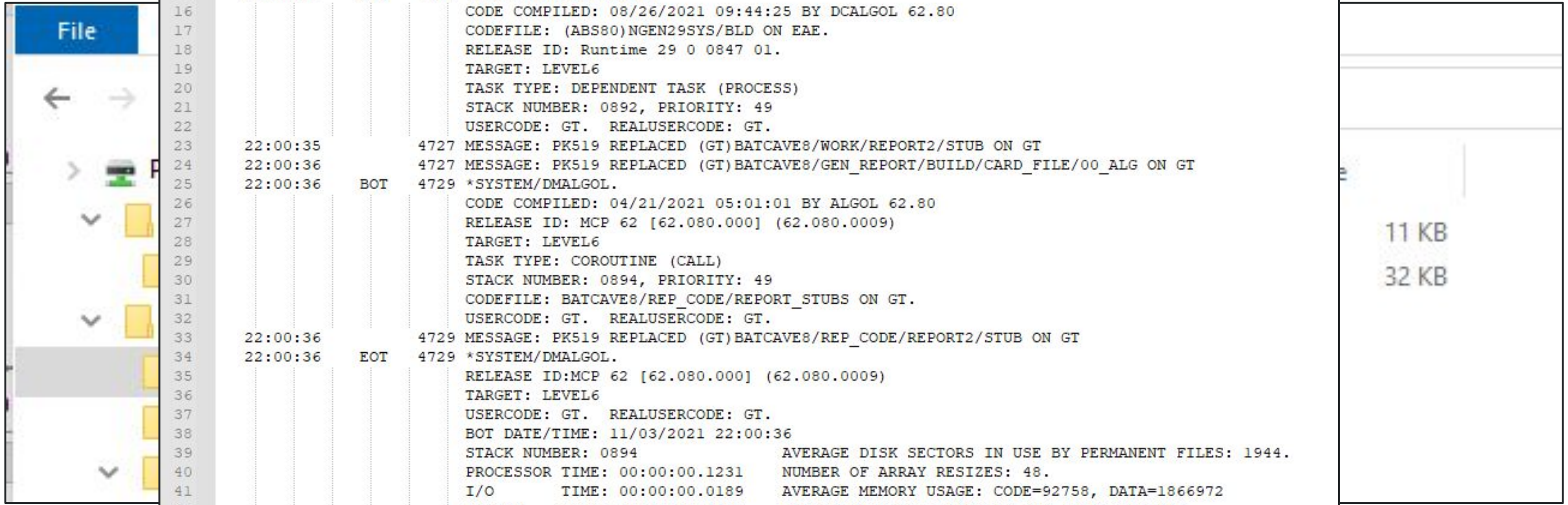
- Batch results in multiple different outputs
  - Print files, Extract files and Database updates
- So how to compare?
  - Lots of Tools for Windows will help
    - Tools like Beyond Compare by Scooter Software can compare folders
      - Options to filter and ignore patterns i.e. exclude dates etc.
    - Use SQL Scripts to compare database records
      - Microsoft SQL Server Management Studio, ideal tool for this
- Not using Windows? - not a problem
  - MCP offers standard capability to:
    - Map drives to MCP locations
    - Convert Printer Backup files to text
    - View DMSII Database structure via SQL Server linked databases

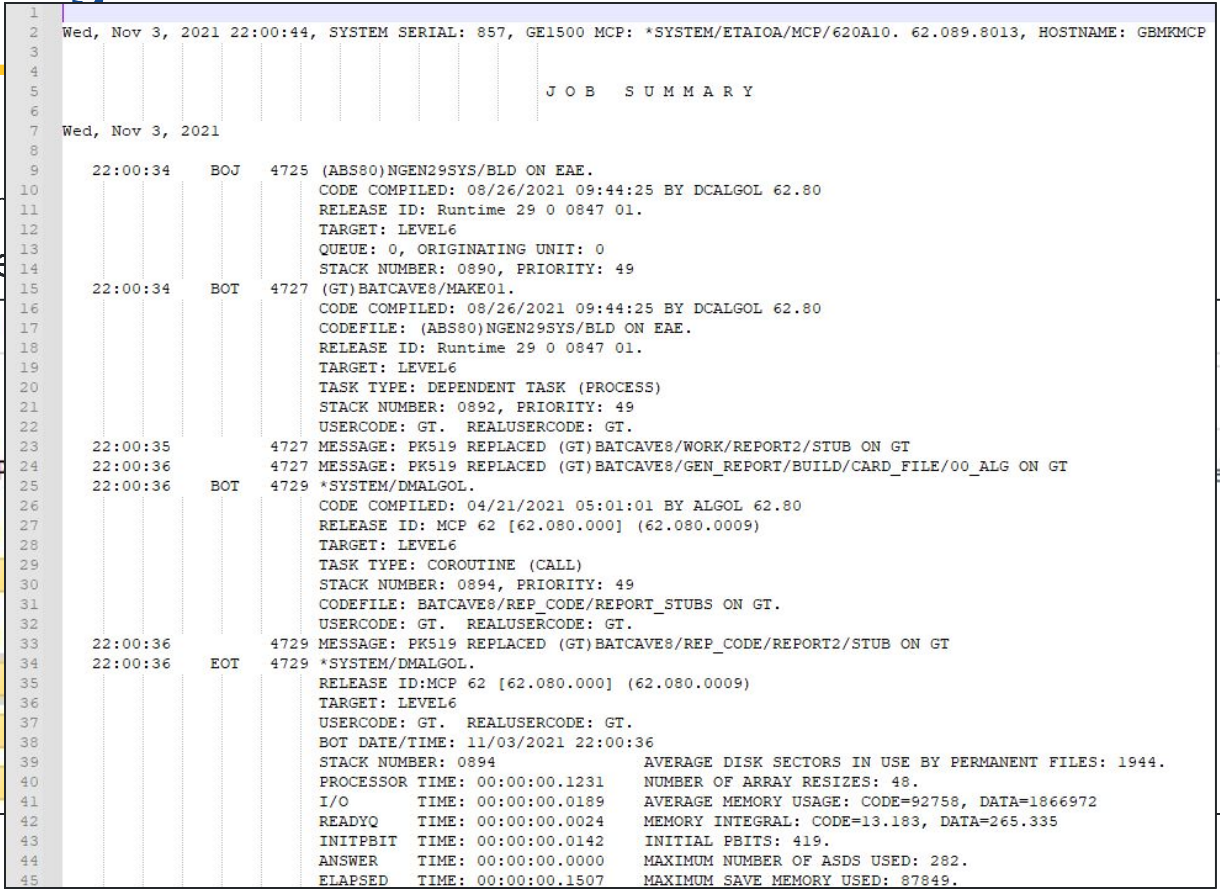
Exposing MCP to Windows - Files

- The Cl… shares

  Down…

  - For e…

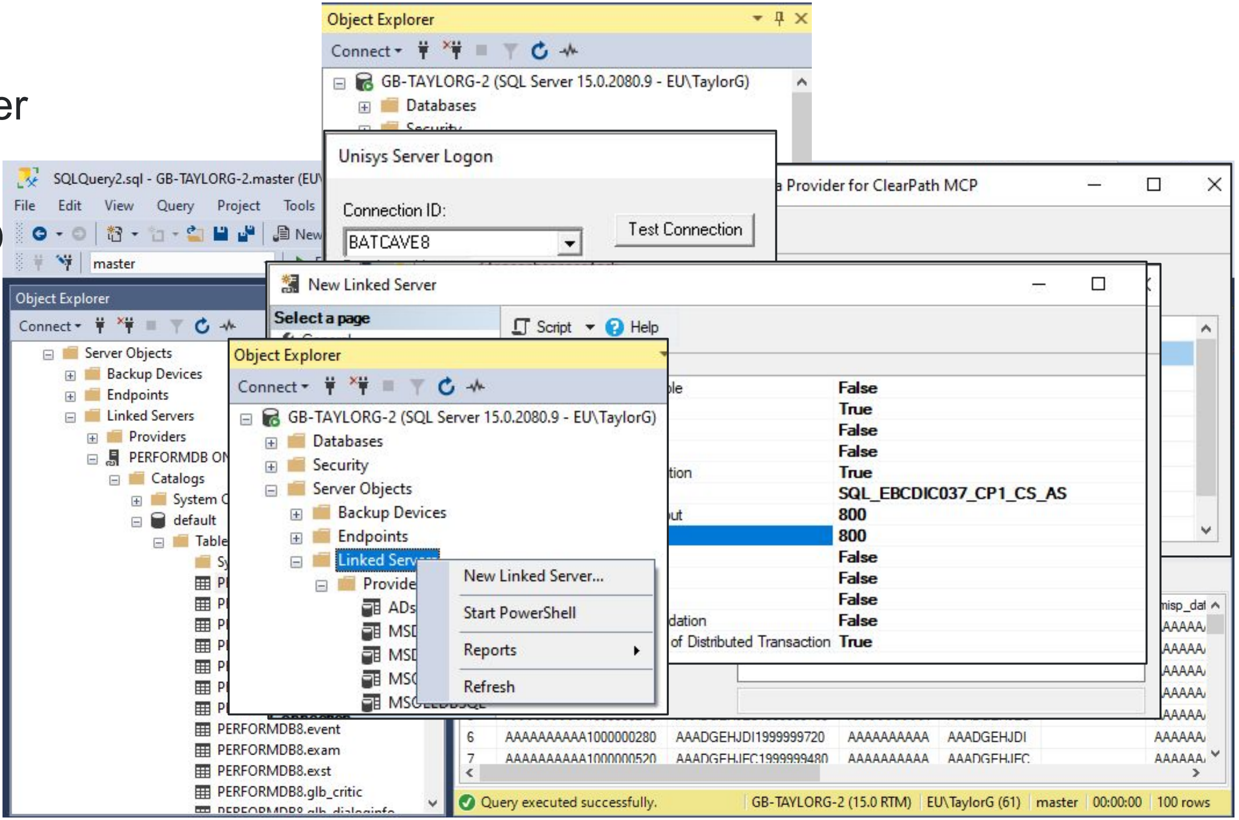




```

Wed, Nov 3, 2021 22:00:44, SYSTEM SERIAL: 857, GE1500 MCP: *SYSTEM/ETAIOA/MCP/620A10. 62.089.8013, HOSTNAME: GBMKMCP


                                                  J O B   S U M M A R Y

Wed, Nov 3, 2021

   22:00:34    BOJ   4725 (ABS80)NGEN29SYS/BLD ON EAE.
                          CODE COMPILED: 08/26/2021 09:44:25 BY DCALGOL 62.80
                          RELEASE ID: Runtime 29 0 0847 01.
                          TARGET: LEVEL6
                          QUEUE: 0, ORIGINATING UNIT: 0
                          STACK NUMBER: 0890, PRIORITY: 49
   22:00:34    BOT   4727 (GT)BATCAVE8/MAKE01.
                          CODE COMPILED: 08/26/2021 09:44:25 BY DCALGOL 62.80
                          CODEFILE: (ABS80)NGEN29SYS/BLD ON EAE.
                          RELEASE ID: Runtime 29 0 0847 01.
                          TARGET: LEVEL6
                          TASK TYPE: DEPENDENT TASK (PROCESS)
                          STACK NUMBER: 0892, PRIORITY: 49
                          USERCODE: GT.  REALUSERCODE: GT.
   22:00:35          4727 MESSAGE: PK519 REPLACED (GT)BATCAVE8/WORK/REPORT2/STUB ON GT
   22:00:36          4727 MESSAGE: PK519 REPLACED (GT)BATCAVE8/GEN_REPORT/BUILD/CARD_FILE/00_ALG ON GT
   22:00:36    BOT   4729 *SYSTEM/DMALGOL.
                          CODE COMPILED: 04/21/2021 05:01:01 BY ALGOL 62.80
                          RELEASE ID: MCP 62 [62.080.000] (62.080.0009)
                          TARGET: LEVEL6
                          TASK TYPE: COROUTINE (CALL)
                          STACK NUMBER: 0894, PRIORITY: 49
                          CODEFILE: BATCAVE8/REP_CODE/REPORT_STUBS ON GT.
                          USERCODE: GT.  REALUSERCODE: GT.
   22:00:36          4729 MESSAGE: PK519 REPLACED (GT)BATCAVE8/REP_CODE/REPORT2/STUB ON GT
   22:00:36    EOT   4729 *SYSTEM/DMALGOL.
                          RELEASE ID:MCP 62 [62.080.000] (62.080.0009)
                          TARGET: LEVEL6
                          USERCODE: GT.  REALUSERCODE: GT.
                          BOT DATE/TIME: 11/03/2021 22:00:36
                          STACK NUMBER: 0894                AVERAGE DISK SECTORS IN USE BY PERMANENT FILES: 1944.
                          PROCESSOR TIME: 00:00:00.1231     NUMBER OF ARRAY RESIZES: 48.
                          I/O       TIME: 00:00:00.0189     AVERAGE MEMORY USAGE: CODE=92758, DATA=1866972
                          READYQ    TIME: 00:00:00.0024     MEMORY INTEGRAL: CODE=13.183, DATA=265.335
                          INITPBIT  TIME: 00:00:00.0142     INITIAL PBITS: 419.
                          ANSWER    TIME: 00:00:00.0000     MAXIMUM NUMBER OF ASDS USED: 282.
                          ELAPSED   TIME: 00:00:00.1507     MAXIMUM SAVE MEMORY USED: 87849.
```

Exposing MCP to Windows - Linked Database

1. Install MCP OLEDB provider
   Download from \\<MCPHost>\INSTALLS
2. Create a Data Source for D
   - Use Test Connection Utility
3. Set Options against Unisys provider in SQL Server
4. Create new linked server



ble	False
	True
	False
	False
tion	True
	SQL_EBCDIC037_CP1_CS_AS
ut	800
	800
	False
	False
	False
dation	False
of Distributed Transaction	True

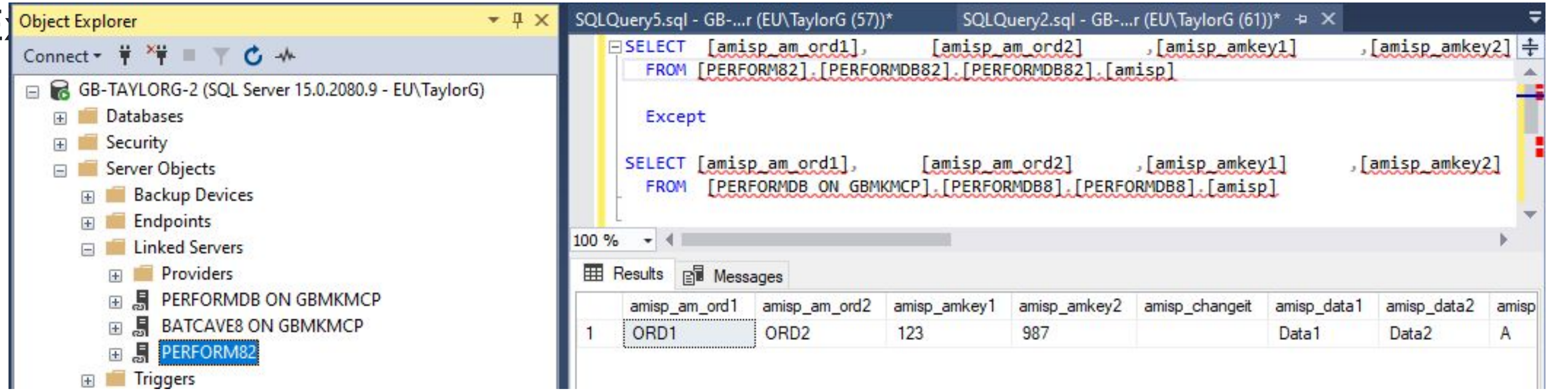
```
6   AAAAAAAAAA1000000280  AAADGEHJDI1999999720  AAAAAAAAAA  AAADGEHJDI  AAAAAA
7   AAAAAAAAAA1000000520  AAADGEHJEC1999999480  AAAAAAAAAA  AAADGEHJEC  AAAAAA
```

Query executed successfully. | GB-TAYLORG-2 (15.0 RTM) | EU\TaylorG (61) | master | 00:00:00 | 100 rows

|

Comparing Two database tables

- The SQL language has some basic Set comparison syntax
  - A INTERSECT B – The members **A** & **B** in Common
  - A EXCEPT B – Members of **A** not in **B**
- This provides simple checks to find differences between two tables
  - Works with Linked Databases
  - E



```
SELECT  [amisp_am_ord1],      [amisp_am_ord2]      ,[amisp_amkey1]      ,[amisp_amkey2]
  FROM [PERFORM82].[PERFORMDB82].[PERFORMDB82].[amisp]

  Except

SELECT [amisp_am_ord1],      [amisp_am_ord2]      ,[amisp_amkey1]      ,[amisp_amkey2]
  FROM  [PERFORMDB ON GBMKMCP].[PERFORMDB8].[PERFORMDB8].[amisp]
```

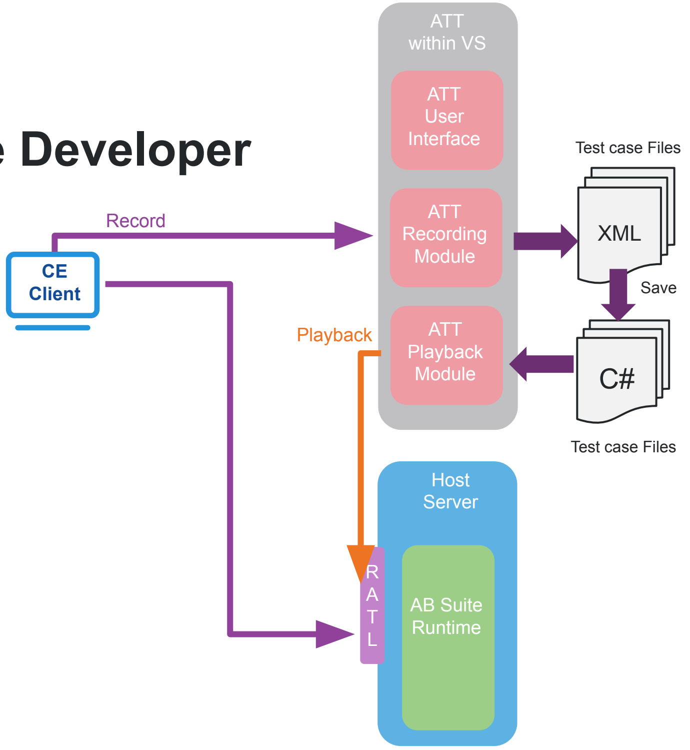
	amisp_am_ord1	amisp_am_ord2	amisp_amkey1	amisp_amkey2	amisp_changeit	amisp_data1	amisp_data2	amisp
1	ORD1	ORD2	123	987		Data1	Data2	A

Securing Your Tomorrow®

Automated Test Tool (ATT)

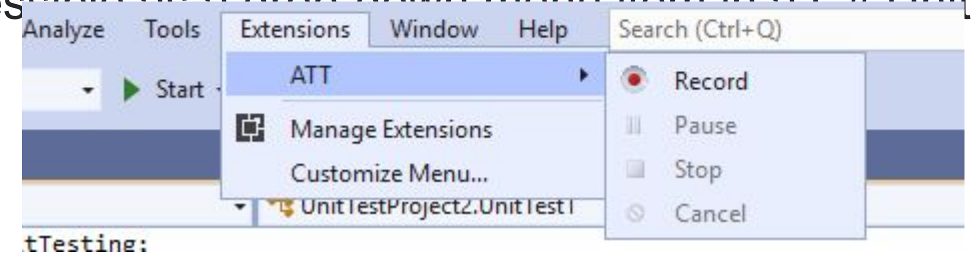
What is ATT?

- **Inbuilt testing Tool supplied as part of AB Suite Developer**
  - Supports all AB Suite Runtime Environments
  - Works with Debugger via RATL
- Provides ability:
  - Record sessions from Component Enabler Clients
    - Once recorded generates standard C# Unit Tests
  - Playback Direct into Runtime
    - Responses validated against definable rules
  - Ability to edit expected return values
  - Can edit generated C# Unit Test
    - Allows injection of additional data



What is ATT? cont

- Leverages Visual Studio Test environment and framework
  - ATT tests (.smtest) created as part of a standard C# Unit Test project
  - Results window and reporting consistent with other Visual Studio test types
- Accessible as a drop down menu item in a C# Unit Test project



- Execution can be automated as part of Pipelines in Azure DevOps

ATT Use – Process overview

1. Deploy Runtime Application
2. Create a “C# Unit Test” Visual Studio Project
3. Start ATT Recording via ATT menu in Visual Studio Project
   - Configure port number ATT listens on – default 8888
4. Start CE Client enabling ATT connection
5. Step through Screens as per normal using CE Client
6. When test sequence complete stop ATT recording
   - This will automatically create the C# test project files to enable this recording to be played back
7. Configure Unit Test Project with connection details of target Runtime
8. Run Test project to play back recorded test
9. Check results

Securing Your Tomorrow®

Further Options

What are your other testing options?

- Component Enabler gateway to building your own custom test capabilities
  - Provides a simple, supported programmatic interface into your Applications
    - Standard Generators provide simple interfaces for use with other tools
  - Approach Chosen by Unisys for our own test tools
    - ATT
    - As previously mentioned the “Test Gateway” **prototype**
- Other Options?
  - **B**usiness **A**pplication **T**est **Man**agement (**BATMan)** another potential option
    - Extracts test cases from your own Log files
    - Ideal for regression testing i.e. Test once replay many
    - Own Replay engine – uses same RATL interface as Component Enabler

So what is *BATMan* ?

- ***BATMan*** is a process and toolset to help deliver Automated online testing
- The toolset currently consist of three main components

  BATCAVE
  - (***B***usiness ***A***pplication Test ***C***apture ***A***nd ***V***erification ***E***ngine)
  - This is used to store and compare transactions

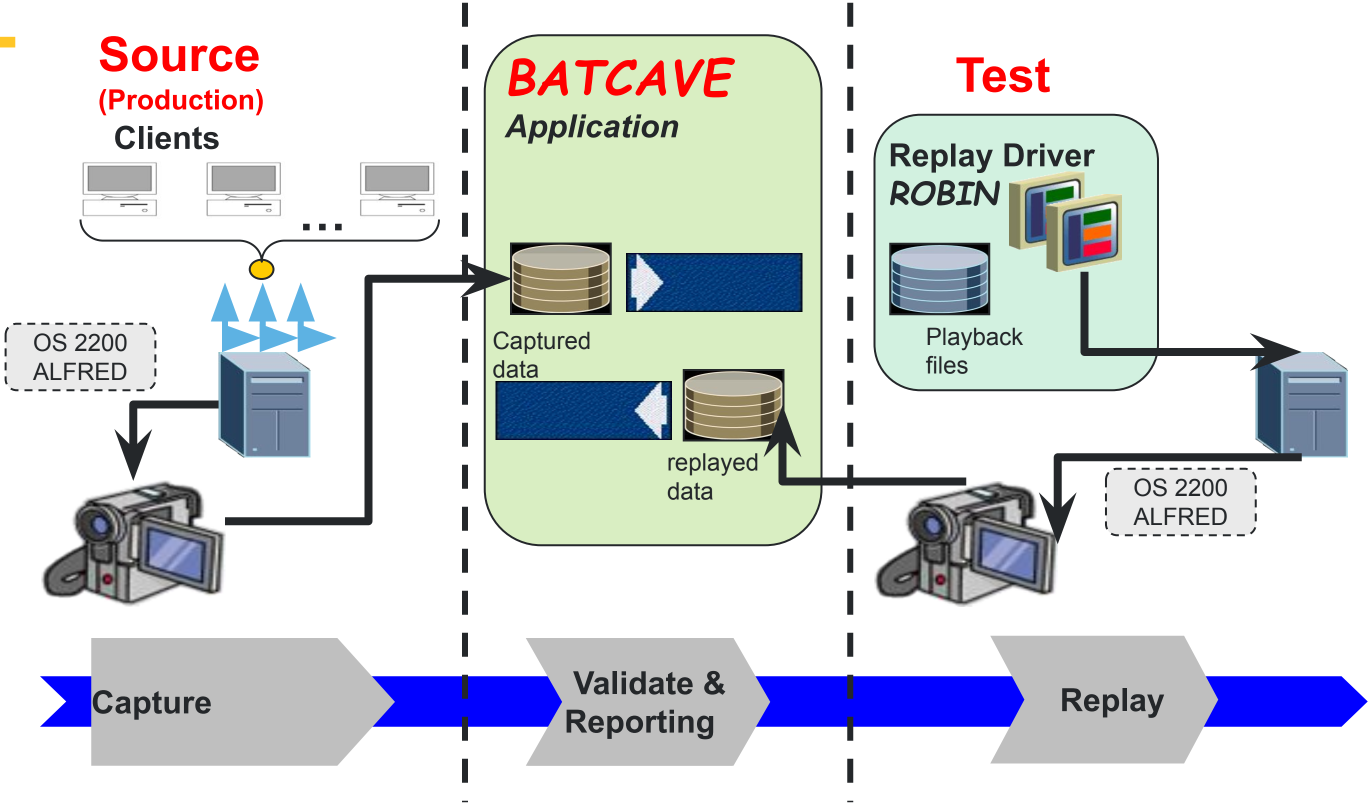
  ROBIN
  - (***R***ealistic ***O***nline ***B***atch ***In***put)
  - This is the default transaction replay driver

  ALFRED
  - (***A***scii ***L***ogging ***F***or ***R***ecording of ***E***AE ***D***ata)
  - This is tool for capturing log like data on the OS 2200 platform

How does it work?

- Rather than defining a test and it's success criteria. ***BATMan***, records a session against one Application/System and then uses the inputs to play those same transactions against a test Application/System. The results/responses of the first session are then compared against the target system responses
- So …
  - ***BATMan*** captures real transactions
    - That includes the mistakes and miss keys that Humans always make
  - If you capture live transactions then you test what you are using not what you think you need to test
    - Real versus synthetic transactions
- This is ideal for testing:
    - Software (EAE/AB Suite IC) upgrades
    - New Hardware
    - Moving to AB Suite
- Not limited to GUI screen input
  - LINC/Transaction Logs used as input allows transaction data from Character mode sessions as well
    - Reply via RATL

BATMan Workflow



Powered by AB Suite!

- The main component of ***BATMan*** is the ***BATCAVE*** application
- Originally developed in EAE but has been subsequently upgraded to AB Suite
  - Allows system to run on any AB Suite platform!
  - Makes use of ALGOL Libraries, VB scripts and ‘C’ libraries depending on platform.
    - Libraries built on the fly by reports so no separate files to maintain
- Use of AB Suite makes tool very easy to extend
  - For Example : Don’t want to use ROBIN to playback?
    - Data stored in AB Suite Runtime DB
    - Easy to extend to create Input data for other tools i.e. Postman input

Notice on Copyright

- *BATMan* has nothing to do with a Comic book hero!
  - Oxford Dictionary definition of "Batman" is an English Army Captains butler
    - In my story the English Army captains name was *ROBIN*
    - The butlers name was *ALFRED*
- *BATCAVE* comes from Cave's that hold Bats
    - They tend to be full of Guano (Bat droppings) which is used as fertiliser to cultivate new plants from seed
    - Caves were also used in ancient times to store things
  - So the analogy is that the *BATCAVE* is used to grow and store your test scripts

But if you have any other ideas then please send them to Warner Brothers!!!!

Thank You