



## Things you can do in AB Suite but not in EAE

Andy Wardle | Senior Architect



## Developer Usability

# Developer Usability

---

- You can change an element's type
  - Change a Standard Component to a Memo Component, or a Memo to an Event
  - Change an Ispec SD to a database item
  - No more delete and recreate!

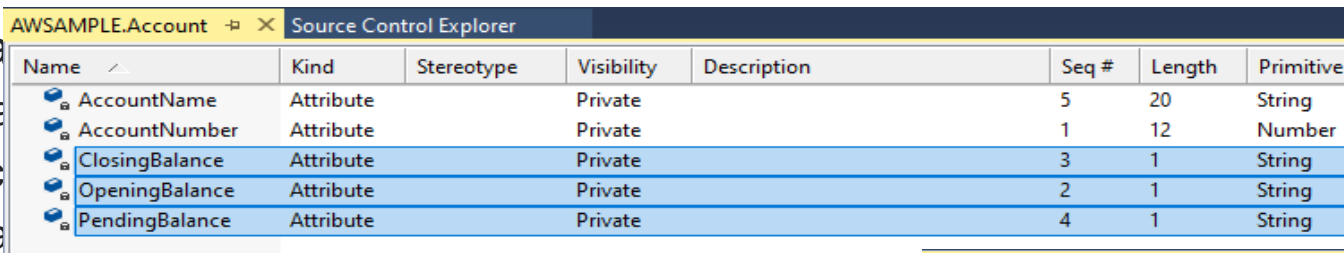
# Developer Usability

---

- You can change an element's type
  - Change a Standard Component to a Memo Component, or a Memo to an Event
  - Change an Ispec SD to a database item
  - No more delete and recreate!
- Bulk changes to properties of multiple attributes
  - e.g. length from 10 to 12, number to signed, decimals from 0 to 2

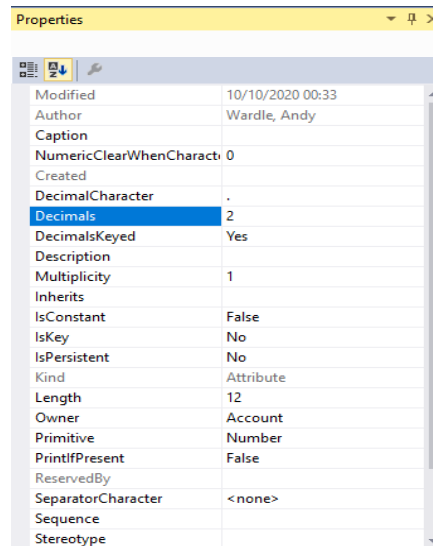
# Developer Usability

- You can change a
  - Change a Standard
  - Change an Spec
  - No more delete a
- Bulk changes to properties of multiple attributes
  - e.g. length from 10 to 12, number to signed, decimals from 0 to 2



AWSAMPLE.Account Source Control Explorer

Name	Kind	Stereotype	Visibility	Description	Seq #	Length	Primitive
AccountName	Attribute		Private		5	20	String
AccountNumber	Attribute		Private		1	12	Number
ClosingBalance	Attribute		Private		3	1	String
OpeningBalance	Attribute		Private		2	1	String
PendingBalance	Attribute		Private		4	1	String



Properties

Modified	10/10/2020 00:33
Author	Wardle, Andy
Caption	
NumericClearWhenCharact	0
Created	
DecimalCharacter	.
Decimals	2
DecimalsKeyed	Yes
Description	
Multiplicity	1
Inherits	
IsConstant	False
IsKey	No
IsPersistent	No
Kind	Attribute
Length	12
Owner	Account
Primitive	Number
PrintIfPresent	False
ReservedBy	
SeparatorCharacter	<none>
Sequence	
Stereotype	

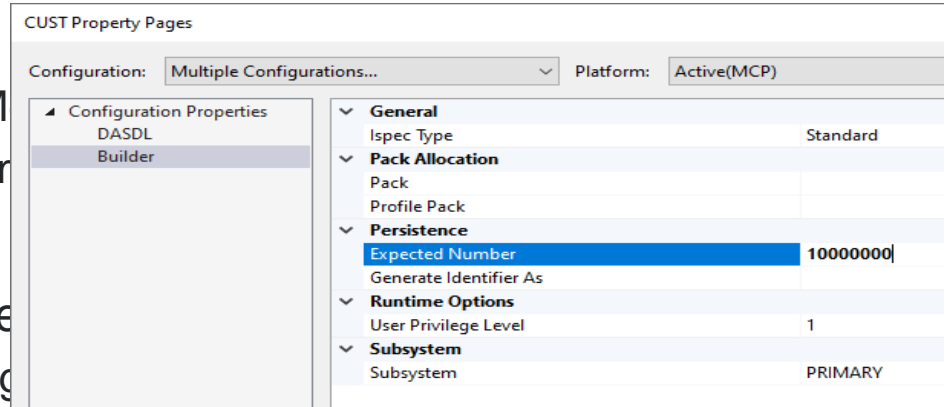
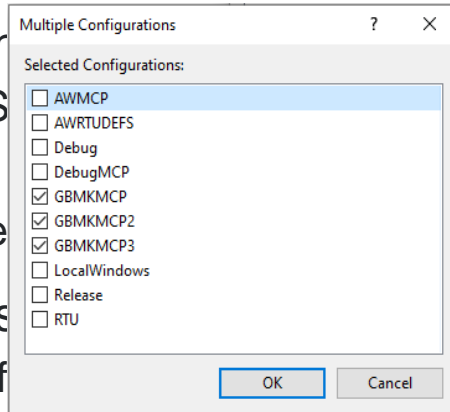
# Developer Usability

---

- You can change an element's type
  - Change a Standard Component to a Memo Component, or a Memo to an Event
  - Change an Ispec SD to a database item
  - No more delete and recreate!
- Bulk changes to properties of multiple attributes
  - e.g. length from 10 to 12, number to signed, decimals from 0 to 2
- Bulk changes to Configuration properties
  - Multiple Configurations for the same element
  - Multiple elements in the same Configuration
  - Multiple elements in multiple Configurations

# Developer Usability

- You can change a configuration
  - Change a configuration
  - Change an element
  - No more developer iterations
- Bulk changes to Configuration properties
  - e.g. length for multiple configurations
- Bulk changes to Configuration properties
  - Multiple Configurations for the same element
  - Multiple elements in the same Configuration
  - Multiple elements in multiple Configurations



# Developer Usability

---

- You can change an element's type
  - Change a Standard Component to a Memo Component, or a Memo to an Event
  - Change an Ispec SD to a database item
  - No more delete and recreate!
- Bulk changes to properties of multiple attributes
  - e.g. length from 10 to 12, number to signed, decimals from 0 to 2
- Bulk changes to Configuration properties
  - Multiple Configurations for the same element
  - Multiple elements in the same Configuration
  - Multiple elements in multiple Configurations
- Default limit of 200 bulk elements, developers can choose to increase this limit



# Developer Usability

---

- Editor line length has no practical limit
  - Use the Continuation Character (\) to break lines as required
  - Can be used wherever there is a blank space

# Developer Usability

---

- Editor line length has no practical limit
  - Use the Continuation Character (\) to break lines as required
  - Can be used wherever there is a blank space
- Developer is Case Insensitive
  - You can use mixed case for element names and in LDL+ statements
  - No automatic conversion to upper case, unlike EAE

# Developer Usability

---

- Editor line length has no practical limit
  - Use the Continuation Character (\) to break lines as required
  - Can be used wherever there is a blank space
- Developer is Case Insensitive
  - You can use mixed case for element names and in LDL+ statements
  - No automatic conversion to upper case, unlike EAE
- Developers can choose their preferred format for commands via an Editor option
  - Upper case (LOOKUP), lower case (lookup), Camel case (LookUp), abbreviated (LU/lu)

# Developer Usability

- Editor line length has no practical limit
  - Use the Continuation Character (\) to break lines as required
  - Can be used wherever there is a blank space
- Developer is Case Insensitive
  - You can use mixed case for element names and in LDL+ statements
  - No automatic conversion to upper case, unlike EAE

```
⊞ Determine From CUST.CUSTNAMES (" ") Serial Secure Gs GLB.STATUS           : This is an extremely long comment to demonstrate no practical limit to a logic line length
⊞ End

⊞ Determine From CUST.CUSTNAMES (" ") \
    Serial \
    Secure \
    Gs \
    GLB.STATUS                       : This is the same statement separated over multiple lines using the Continuation Character
⊞ End

lu    CUSTOMER CUST
LU    CUSTOMER CUST
LOOKUP CUSTOMER CUST
lookUp CUSTOMER CUST
```

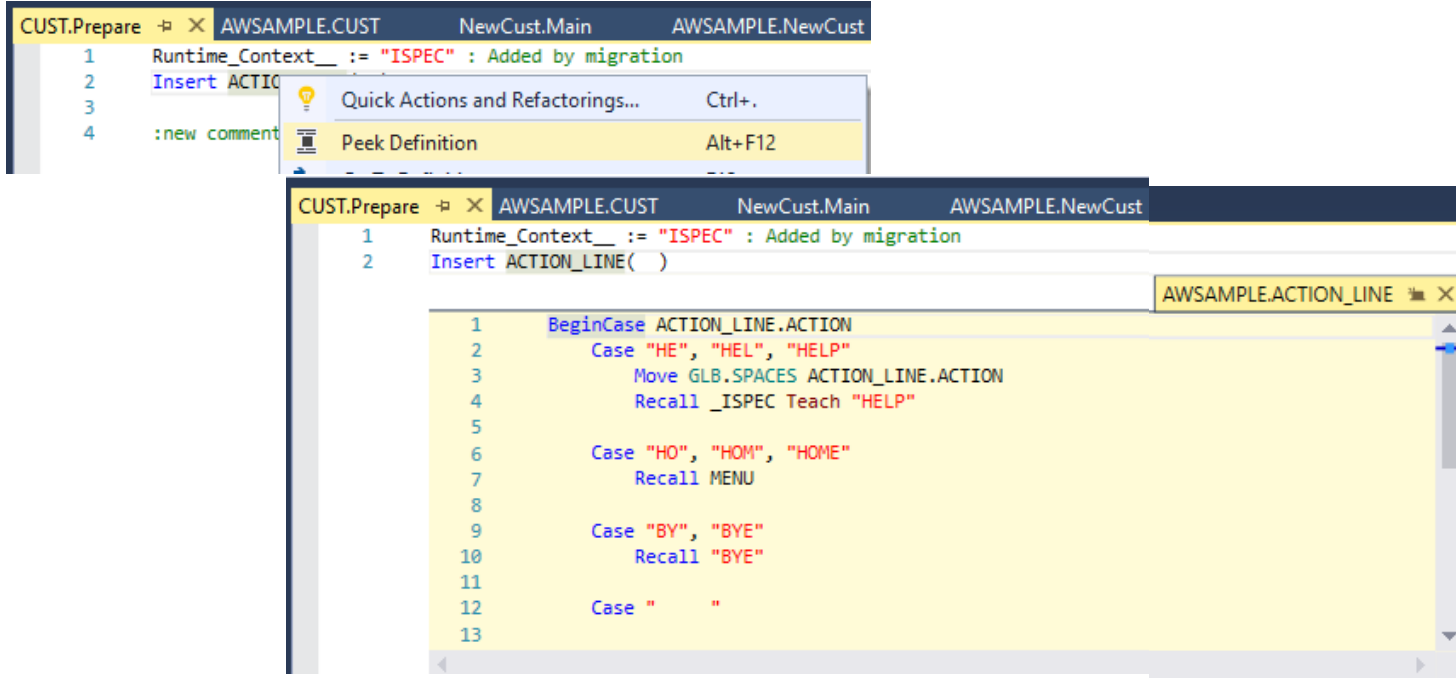
# Developer Usability

---

- Peek allows method logic to be viewed/edited where its invoked

# Developer Usability

- Peek allows method logic to be viewed/edited where its invoked



# Developer Usability

---

- Peek allows method logic to be viewed/edited where its invoked
- Document Tab provides flexible views via column sorts/selections/filters

# Developer Usability

- Peek allows method logic to be viewed/edited where its invoked
- Document Tab provides flexible views via column sorts/selections/filters

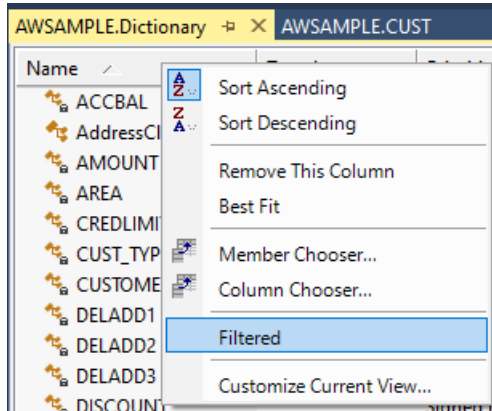
Name	Template	Primitive	Decimals	Length
ACCBAL		Signed Number	2	10
AddressClass		Class		102
AMOUNT		Signed Number	2	8
AREA		String		30
CREDLIMIT		Number	0	6
CUST_TYPE		String		1
CUSTOMER		String		6
DELADD1		String		30
DELADD2		String		30
DELADD3		String		30
DISCOUNT		Signed Number	2	6
DISCPC		Number	0	2
DOCUMENT		Number	0	8
INVOICED		String		1

Name	Template	Primitive	Decimals	Length
AddressClass		Class		102
PERIOD		Number	0	4
DISCPC		Number	0	2
DOCUMENT		Number	0	8
REORDLEV		Number	0	6
SELLPRICE		Number	4	8
CREDLIMIT		Number	0	6
TRANDATE		Number	0	5
TRANTIME		Number	0	8
DISCOUNT		Signed Number	2	6
AMOUNT		Signed Number	2	8



# Developer Usability

- Peek allows method logic to be viewed/edited where its invoked
- Document Tab provides flexible views via column sorts/selections/filters



Name	Template	Primitive	Decimals	Length
P	Enter text h...	Enter text h...	Ent...	Ent...
PERIOD		Number	0	4
POSTADD1		String		30
POSTADD2		String		30
POSTADD3		String		30
PRODUCT		String		6

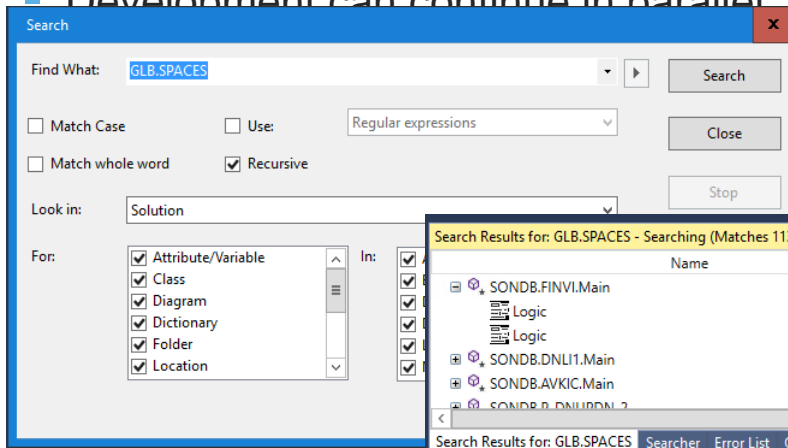
# Developer Usability

---

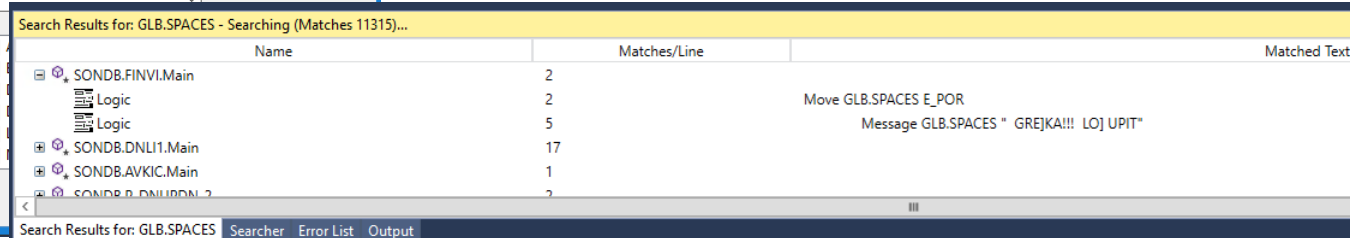
- Peek allows method logic to be viewed/edited where its invoked
- Document Tab provides flexible views via column sorts/selections/filters
- Each search runs as an independent thread
  - Multiple searches in parallel
  - Development can continue in parallel
  - As soon as references appear in search window you can double click to review them

# Developer Usability

- Peek allows method logic to be viewed/edited where its invoked
- Document Tab provides flexible views via column sorts/selections/filters
- Each search runs as an independent thread
  - Multiple searches in parallel
  - Development can continue in parallel



In this window you can double click to review them





Language

# Language

---

- Complex expressions in logic statements

# Language

---

- Complex expressions in logic statements

```
EAE:  MV; CURRBAL      GD-CURRBAL
      MV; INTRATE     GD-INTRATE
      INS; GP-CALCINT : Result in GD-CALCRSLT
      AD; (24)        GD-AMOUNT
      MV; (7)         GD-INDEX
      COMP;           ANUMBER (NUMARRAY * GD-AMOUNT * GD-CALCRSLT)
```

```
AB Suite: ANumber := NumberArray [7] * (Amount + 24) * CalculateInterest(CurrentBalance,InterestRate)
```

# Language

- Complex expressions in logic statements

**EAE:**

```
MV; CURRBAL      GD-CURRBAL
MV; INTRATE      GD-INTRATE
INS; GP-CALCINT   : Result in GD-CALCRSLT
AD; (24)         GD-AMOUNT
MV; (7)          GD-INDEX
COMP;           ANUMBER (NUMARRAY * GD-AMOUNT * GD-CALCRSLT)
```

**AB Suite:** ANumber := NumberArray [7] \* (Amount + 24) \* CalculateInterest(CurrentBalance,InterestRate)

**EAE:**

```
MV; CURRBAL      GD-CURRBAL
MV; INTRATE      GD-INTRATE
INS; GP-CALCINT   : Result in GD-CALCRSLT
MV; GD-CALCRSLT ANUMBER
```

**AB Suite:** Move CalculateInterest(CurrentBalance,InterestRate) ANumber

# Language

---

- Complex expressions in logic statements

```
EAE:      MV; (Error)                GD-MSG
          ATS; ANUMBER                GD-MSG
          ATS; (- Contact Technical Support" GD-MSG
          ME; ERROR                    GD-MSG
```

```
AB Suite: Message Error "Error" &+ ANumber &+ " - Contact Technical Support"
```



# Language

- Complex expressions in logic statements

```
EAE:      MV; (Error)                GD-MSG
          ATS; ANUMBER                GD-MSG
          ATS; (- Contact Technical Support" GD-MSG
          ME; ERROR                    GD-MSG
```

```
AB Suite: Message Error "Error" &+ ANumber &+ " - Contact Technical Support"
```

```
EAE:      MU; SALARY INCREMENT GIVING;      GD-INC
          DW; GD-INC > MAXRISE
          MV; (Pay Rise Too big = )         GD-MSG
          ATS; GD-INC                        GD-MSG
          ME; ATTENTION                      GD-MSG
END:
⊖If (Salary * Increment) > MaxPayRiseAllowed
    Message Attention "Pay Rise Too Big =" &+ (Salary * Increment)
End
```

# Language

- Complex expressions in logic statements

```
EAE:      MV; (Error)                GD-MSG
          ATS; ANUMBER                GD-MSG
          ATS; (- Contact Technical Support" GD-MSG
          ME; ERROR                    GD-MSG
```

AB Suite: Message Error "Error" &+ ANumber &+ " - Contact Technical Support"

```
EAE:      MU; SALARY INCREMENT GIVING;    GD-INC
          DW; GD-INC > MAXRISE
          MV; (Pay Rise Too big = )       GD-MSG
          ATS; GD-INC                      GD-MSG
          ME; ATTENTION                    GD-MSG
          END;
```

AB Suite: `⊖If (Salary * Increment) > MaxPayRiseAllowed`  
          Message Attention "Pay Rise Too Big =" &+ (Salary \* Increment)  
          End

# Language

- Complex expressions in logic statements

```
EAE:      MV; (Error)                GD-MSG
          ATS; ANUMBER                GD-MSG
          ATS; (- Contact Technical Support" GD-MSG
          ME; ERROR                    GD-MSG
```

AB Suite: Message Error "Error" &+ ANumber &+ " - Contact Technical Support"

```
EAE:      MU; SALARY INCREMENT GIVING;    GD-INC
          DW; GD-INC > MAXRISE
          MV; (Pay Rise Too big = )      GD-MSG
          ATS; GD-INC                    GD-MSG
          ME; ATTENTION                  GD-MSG
          END;
```

AB Suite: `⊖ If (Salary * Increment) > MaxPayRiseAllowed`  
          Message Attention "Pay Rise Too Big =" &+ (Salary \* Increment)  
          End

- Fewer lines of code and fewer variables, compared to EAE, for the same outcome

# Language

---

- ForEach – a new looping verb with two variants
  - ForEach <Ispec Instance/Record> in <Ispec>/<Profile>/<Extract File>
    - Alternative to looping Determine and LookUp commands
  - ForEach <Array Element> in <Array>
    - New way to iterate over arrays without needing to check the upper bounds

# Language

- ForEach – a new looping verb with two variants
  - ForEach <Ispec Instance/Record> in <Ispec>/<Profile>/<Extract File>
    - Alternative to looping Determine and LookUp commands
  - ForEach <Array Element> in <Array>
    - New way to iterate over arrays without needing to check the upper bounds

```
ForEach CUST In CUST.CUSTNAMES
    Message Attention CUST.NAM &+ "is the Customer Name and" &+ CUST.CUSTOMER &+ "is the Customer ID"
End
```

```
ForEach Name In CustomerNamesArray
    Message Attention "The Customer Name is" &+ Name
End
```



## Model Design

# Model Design

---

- Genuine Constants
  - Any attribute (variable) can be defined with an Initial Value, like EAE
  - But now you can mark the variable as “IsConstant”, and the Editor will syntax any attempts in LDL+ to assign a value to the variable

# Model Design

- Genuine Constants

- Any attribute (variable) can be defined with an Initial Value, like EAE
- But now you can mark the variable as “IsConstant”, and the Editor will syntax any attempts in LDL+ to assign a value to the variable

Move 25 DMVDefaultRecSize

🔍 (attribute) Signed Number (Length: 11) DMVDefaultRecSize

Operand DMVDefaultRecSize is Read-only (Property IsConstant = True)



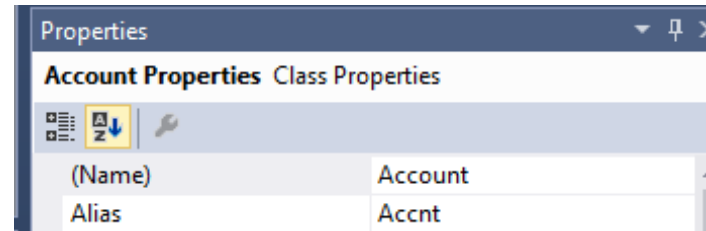
# Model Design

---

- Genuine Constants
  - Any attribute (variable) can be defined with an Initial Value, like EAE
  - But now you can mark the variable as “IsConstant”, and the Editor will syntax any attempts in LDL+ to assign a value to the variable
- Meaningful Names
  - The EAE name limitations no longer exist within Developer
  - Elements have an Alias property that is used to adhere to external restrictions
    - e.g. Database schema elements, Ispecs, Reserved Words

# Model Design

- Genuine Constants
  - Any attribute (variable) can be defined with an Initial Value, like EAE
  - But now you can mark the variable as “IsConstant”, and the Editor will syntax any attempts in LDL+ to assign a value to the variable
- Meaningful Names
  - The EAE name limitations no longer exist within Developer
  - Elements have an Alias property that is used to adhere to external restrictions
    - e.g. Database schema elements, Ispecs, Reserved Words



# Model Design

---

- A new data type – Boolean
- Reports can now have more than 99 Frames/Methods
  - Frames now have proper names, not simply a number
- Reports can now have more than 26 Extract/Print files
  - Extract and Print files now have proper names, not simply a letter
- Extract files can now be used in Ispecs
- You can have as many folder levels as you like
  - Not just the Functional Areas, Activities, Bundles and Report Generate Groups in EAE

# Model Design

---

- Alphanumeric items (“Strings”) can be much larger
  - 262 KB vs 65 KB in EAE
  - Therefore Arrays can be much larger as well

# Model Design

---

- Alphanumeric items (“Strings”) can be much larger
  - 262 KB vs 65 KB in EAE
  - Therefore Arrays can be much larger as well
- Arrays can be defined within Groups

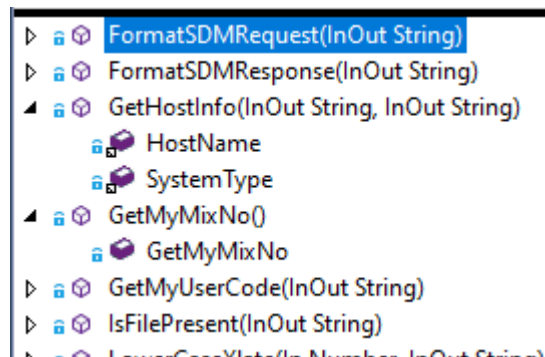
# Model Design

---

- Alphanumeric items (“Strings”) can be much larger
  - 262 KB vs 65 KB in EAE
  - Therefore Arrays can be much larger as well
- Arrays can be defined within Groups
- External methods can be called using zero or more parameters
  - No longer limited to using Glb.Param, which has a maximum length of 262 KB
  - Parameters can be of different types, and much larger in size
  - A result value can also be returned

# Model Design

- Alphanumeric items (“Strings”) can be much larger
  - 262 KB vs 65 KB in EAE
  - Therefore Arrays can be much larger as well
- Arrays can be defined within Groups
- External methods can be called using zero or more parameters
  - No longer limited to using Glb.Param, which has a maximum length of 262 KB
  - Parameters can be of different types, and much larger in size
  - A result value can also be returned



# Model Design

---

- Segment-level methods (GLGs) may be marked as “Public”, allowing them to be
  - Invoked via Client Tools as an alternative to Ispecs
  - Called from external programs



# Model Design

---

- Segment-level methods (GLGs) may be marked as “Public”, allowing them to be
  - Invoked via Client Tools as an alternative to Ispecs
  - Called from external programs
- Methods can be parameterised
  - With multiple parameters of different types
    - e.g. String, Number, Boolean, Array

# Model Design

- Segment-level methods (GLGs) may be marked as “Public”, allowing them to be
  - Invoked via Client Tools as an alternative to Ispecs
  - Called from external programs
- Methods can be parameterised
  - With multiple parameters of different types
    - e.g. String, Number, Boolean, Array
- Methods can return a result
  - Similar to typed procedures/functions in other languages
  - Result type can be any data type, e.g. Boolean, String, Number

```
▲ + [lock] Increment(In Number, In Number)
+ [lock] Increment
+ [lock] IncrementValue
+ [lock] InitialValue
```

AWSAMPLE.Increment		AWSAMPLE.Account	Source Cont
1	Add	InitialValue IncrementValue	Giving GLB.TOTAL
2	Move	GLB.TOTAL Increment	

Account.CalculateValue*		AWSAMPLE.Account	Source Co
1	Move	Increment(5,OpeningBalance)	ClosingBalance

# Model Design

---

- Methods with parameters, return values and restricted data visibility offer opportunities for encapsulation
  - Methods as self-contained modules that can be tested in isolation
  - Reduces/eliminates the need for full application regression testing

# Model Design

---

- Methods with parameters, return values and restricted data visibility offer opportunities for encapsulation
  - Methods as self-contained modules that can be tested in isolation
  - Reduces/eliminates the need for full application regression testing
- Ispecs can have additional user-defined Methods
  - Not just Construct (Pre-Screen), Prepare (Pre-LINC), and Main
  - e.g. NewCust has an “AddNewCustomer” method that can be called from other logic to validate the data and add a new record if successful

# Model Design

- Methods with parameters, return values and restricted data visibility offer opportunities for encapsulation
  - Methods as self-contained modules that can be tested in isolation
  - Reduces/eliminates the need for full application regression testing

**Additional user-defined Methods**

```

Main    AWSAMPLE.LOADREPT    CUST.AddNewCustomer
LookUp  CUSTOMER CUST
If GLB.STATUS = "*****" : Not Found, so good to add
  CUST.Initialize()

  Move GLB.ADD      CUST.MAINT
  Move CREDLIMIT   CUST.CREDLIMIT
  Move CUSTOMER    CUST.CUSTOMER
  Move CUST_TYPE   CUST.CUST_TYPE
  Move DELADD1     CUST.DELADD1
  Move DELADD2     CUST.DELADD2
  Move DELADD3     CUST.DELADD3
  Move NAM         CUST.NAM
  Move POSTADD1   CUST.POSTADD1
  Move POSTADD2   CUST.POSTADD2
  Move POSTADD3   CUST.POSTADD3
  Move REPTCODE   CUST.REPTCODE
  Move SALESREP   CUST.SALESREP

  CUST.Store()
  If GLB.STATUS = GLB.SPACES
    AddNewCustomer := True
  End
End

If not CUST.AddNewCustomer("Customer1","1 The Street","AnyTown","My State","A","A",50000,"R1","CUST1")
  Message Error "Add of Customer1 Failed"
End
If not CUST.AddNewCustomer("Customer2","2 The Street","AnyTown","My State","B","B",50000,"R1","CUST2")
  Message Error "Add of Customer2 Failed"
End
If not CUST.AddNewCustomer("Customer3","3 The Street","AnyTown","My State","C","C",50000,"R1","CUST3")
  Message Error "Add of Customer3 Failed"
End
If not CUST.AddNewCustomer("Customer4","4 The Street","AnyTown","My State","C","C",50000,"R1","CUST4")
  Message Error "Add of Customer4 Failed"
End

```

# Model Design

---

- Methods with parameters, return values and restricted data visibility offer opportunities for encapsulation
  - Methods as self-contained modules that can be tested in isolation
  - Reduces/eliminates the need for full application regression testing
- Specs can have additional user-defined Methods
  - Not just Construct (Pre-Screen), Prepare (Pre-LINC), and Main
  - e.g. NewCust has an “AddNewCustomer” method that can be called from other logic to validate the data and add a new record if successful
- Reports can also have additional user-defined Methods (Windows Runtime only)
  - Same selection of parameters and return value types provided
  - Much more flexible than Print.Frame.Logic in EAE or <Frame>.Main() in AB Suite

# Model Design

---

- Multiple instances of an Ispec can be defined and referenced in logic
  - Each instance is a differently named copy of the Ispec
    - The original Ispec is the template for each copy
    - Template updates automatically reflected in each instance
    - Minimal amount of logic required to manage each instance
  - A typical use case might be a bank account with multiple account holders
  - Allows the data for each instance to be held in memory concurrently
  - EAE would need to store copies of data for each instance in separate variables or GSD/SD groups
    - With logic to move each set of GSD/SD/variable information individually
    - Any change in the original Ispec may require related changes to variables and associated logic

# Model Design

- Simple example using multiple instances of an Ispec
  - Customer A is merged with Customer B to create Customer C

```
LU; EXCUSTA (CUST)
MV; CUST.CUSTOMER          SD-CUSTA-CUSTOMER
MV; CUST.NAM               SD-CUSTA-NAM
MV; CUST.POSTADD1         SD-CUSTA-ADD1
MV; CUST.POSTADD2         SD-CUSTA-ADD2
MV; CUST.POSTADD3         SD-CUSTA-ADD3
MV; CUST.CREDLIMIT        SD-CUSTA-CREDLIMIT

LU; EXCUSTB (CUST)
MV; CUST.CUSTOMER          SD-CUSTB-CUSTOMER
MV; CUST.NAM               SD-CUSTB-NAM
MV; CUST.POSTADD1         SD-CUSTB-ADD1
MV; CUST.POSTADD2         SD-CUSTB-ADD2
MV; CUST.POSTADD3         SD-CUSTB-ADD3
MV; CUST.CREDLIMIT        SD-CUSTB-CREDLIMIT

AE; CUST
AU; GLB.ADD                MAINT
AU; SD-MERGE-CUSTOMER     CUSTOMER
AU; SD-CUSTA-NAME         NAM
AU; SD-CUSTA-ADD1         POSTADD1
AU; SD-CUSTA-ADD2         POSTADD2
AU; SD-CUSTA-ADD3         POSTADD3
AD; SD-CUSTA-CREDLIMIT    SD-CUSTB-CREDLIMIT GIV GLB.TOTAL
AU; GLB.TOTAL             CREDLIMIT
AU; WRITE
```

The screenshot shows a software development environment with a class hierarchy table and a code editor. The class hierarchy table lists members of the `CustMerge.Main` class, including `ExistingCustA`, `ExistingCustB`, `ExistingCustomerA`, `ExistingCustomerB`, `MergedCust`, and `MergedCustomer`. The code editor shows the implementation of the `MergedCustomer` class, including an `Initialize()` method that sets up the `MergedCustomer` instance with data from `ExistingCustomerA` and `ExistingCustomerB`.

Name	Kind	Inherits
ExistingCustA	Variable	CUST.CUSTOMER
ExistingCustB	Variable	CUST.CUSTOMER
ExistingCustomerA	Variable	CUST
ExistingCustomerB	Variable	CUST
MergedCust	Variable	CUST.CUSTOMER
MergedCustomer	Variable	CUST

```
Lookup ExistingCustA ExistingCustomerA
Lookup ExistingCustB ExistingCustomerB

MergedCustomer.Initialize()
MergedCustomer := ExistingCustomerA
MergedCustomer.CUSTOMER := MergedCust
MergedCustomer.CREDLIMIT := ExistingCustomerA.CREDLIMIT + ExistingCustomerB.CREDLIMIT
MergedCustomer.Set__Maint("ADD")
MergedCustomer.Store()
```



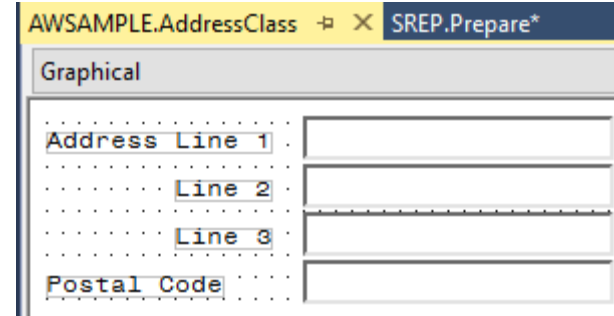
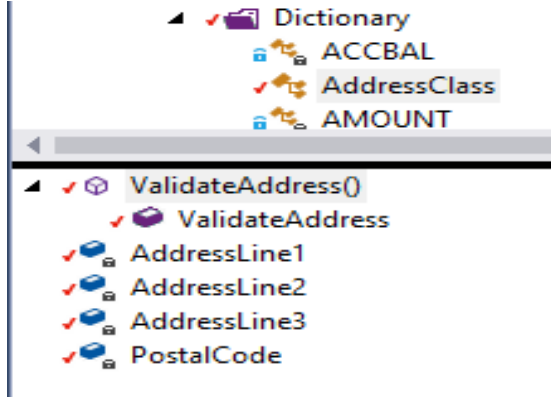
# Model Design

---

- Complex Dictionary Items with ...
  - One or more variables
  - Zero or more painted items
  - Zero or more logic methods

# Model Design

- Complex Dictionary Items with ...
  - One or more variables
  - Zero or more painted items
  - Zero or more logic methods



```
: Returns True if the Address is Invalid
If AddressLine1 = glb.SPACES
    Return TRUE
End

If PostalCode = glb.SPACES
    Return TRUE
End

If PostalCode.GetLength() < 6
    Return TRUE
End
```

# Model Design

---

- Complex Dictionary Items with ...
  - One or more variables
  - Zero or more painted items
  - Zero or more logic methods
- Can be invoked one or more times in other objects using inheritance
  - NB. From 7.0 onwards the terminology changes from Inheritance to Templating

# Model Design

- Complex Dictionary Items with ...
  - One or more variables
  - Zero or more painted items
  - Zero or more logic methods
- Can be invoked one or more times in other objects using inheritance
  - NB. From 7.0 onwards the terminology changes from Inheritance to Templating

Universal Distributors

SALES REPRESENTATIVE COMPONENT

Sales Rep Number

Name

Geographic Area

Asia

Australia/NZ

Europe

North America

South America

Other

Add Inquir First Next Prior Last Change Delete

Home Address Office Address

Address Line 1

Line 2

Line 3

Postal Code

Name	Kind	Inherits
OfficeAddress	Attribute	AddressClass
HomeAddress	Attribute	AddressClass

```
If HomeAddress.ValidateAddress()  
    Message Error "Home Address Invalid"  
End  
If OfficeAddress.ValidateAddress()  
    Message Error "Office Address Invalid"  
End
```



Miscellaneous

# Debugging

---

- Edit and Continue in real time
  - No need to exit debugging, modify code, then restart debugging
- Use of SQL Server enables use of popular SQL Server query products
- Simulates MCP behaviour by using EBCDIC data
  - Database, Extract Files, and Sort collating sequence
  - EBCDIC query/update tool provided
- Debug using your own Client Tools clients via RATL
- Debug individual Public Segment Methods (GLGs)
  - Especially if they are encapsulated

# Debugging

---

- Edit and Continue in real time
  - No need to exit debugging, modify code, then restart debugging
- Use of SQL Server enables use of popular SQL Server query products
- Simulates MCP behaviour by using EBCDIC data
  - Database, Extract Files, and Sort collating sequence
  - EBCDIC query/update tool provided
- Debug using your own Client Tools clients via RATL
- Debug individual Public Segment Methods (GLGs)
  - Especially if they are encapsulated

**\*\* For more on using Debugger attend Gary Taylor's session on Thursday \*\***

# Source Control

---

- You can version almost anything you want to
  - EAE only versioned specific element types
    - Ispecs, Reports, GLGs, Dictionary Items, GSDs, Profiles
    - But, e.g., not Folders, not Generate Sets
    - Versioned elements know nothing of the model's overall hierarchical structure, and nothing about the runtime environment properties
  - With AB Suite you can retrieve an exact copy of the model from Source Control
- Source Control is no longer a proprietary tool (i.e. the Version Control Bank)
  - Now it's Microsoft's Team Foundation Server, a.k.a. Azure DevOps Server/Services
  - And the information is stored in SQL Server databases



# MCP Builder and Runtime

---

- Integrated support for DMS II ReorgDB, enabling 24x7x365 availability
- Integrated support for recent DMS II enhancements, including
  - DMS II Encryption – Ispec Field Level and Audit Trail
  - Granular logging of database accesses
  - Larger data item lengths
  - MigrateDB (dynamic schema changes)
  - Staged Database Reorganisation (almost eliminates limit on Reorg sizes in single Build)
  - Binary Large Objects (BLOBS), e.g. for images, signatures, etc.
- Secure RATL, enabling secure communications with Client Tools clients
- Secure FTP for source file transfer during Builds

# Planned in AB Suite 8.0 ...

---

- OS2200 Runtime, including Secure RATL
- MCP Runtime ...
  - DMS II Structure Level Encryption
  - Report user-defined methods
  - HUB support for 65 KB messages, Business Integrator support for 10 KB messages
- Windows Runtime
  - SQL Server Transparent Data Encryption
  - Secure RATL
- General
  - Maximum length of Alphanumeric non-database variables (incl. Glb.Param) increased to 6 MB

# Thank You

