

The ABL Keeps Getting Better

What's New in the ABL – 11.4

Laura Stern – Principal Software Engineer

Sharanya Sarkar – QA Engineer II

PROGRESS
EXCHANGE 2014

Agenda

- OOABL serialization
- FINALLY block
- GET-CLASS
- JSON Before-Image Support
- 64-bit WebClient

Object Serialization – Motivation

Problem

- There is no standard way to get error information from the AppServer to a client
- There is no way to pass OOABL objects between an ABL client and an AppServer

Solution

Introduce built-in OOABL object serialization

- Works between an ABL client and an AppServer
 - Not Open Client

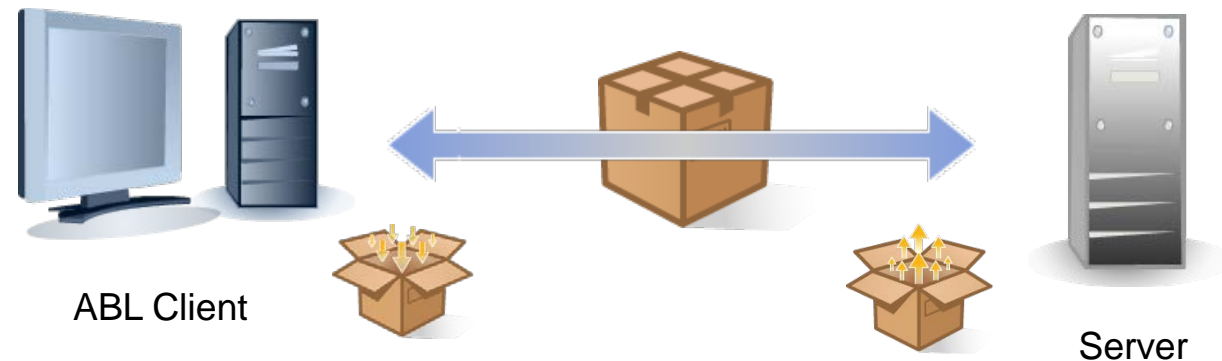
Object Serialization in the ABL

■ Use Cases

- Throwing an error object from the AppServer to an ABL client
- Passing an object between an ABL client and an AppServer
- Passing temp tables that contain ABL object fields between an ABL client and an AppServer

■ Rules for serialization and deserialization

■ Futures Roadmap



Throwing an Error Object – 11.4

```
RETURN ERROR New Progress.Lang.AppError(...).
```

```
ROUTINE-LEVEL ON ERROR UNDO, THROW.
```

```
CATCH err AS Progress.Lang.Error:  
    UNDO, THROW err.  
END.
```



ABL Client



Server

Throwing an Error Object

Pre - 11.4

- Raises ERROR on client
- Generated warning in the AppServer log file
- No object instance returned
- Not even error message available on the client

11.4

- Raises ERROR on client
- Object instance returned
- Error message and all other object data available on the client

What Objects Can You Throw?

- Classes which implement `Progress.Lang.Error`, for example,
 - `Progress.Lang.SysError`
 - `Progress.Lang.AppError`
 - `Progress.Lang.JsonError`
 - `Progress.BPM.BPMError`
- Any user-defined class that implements `Progress.Lang.Error`
 - Typically subclass of `Progress.Lang.AppError`
 - Must be marked `SERIALIZABLE`
- Not `.NET` Exceptions

Error Object – CallStack

- Error objects can contain Callstack information
 - SESSION:ERROR-STACK-TRACE attribute to TRUE
 - -errorstack startup parameter
- Callstack augmented with info from both client and AppServer call stacks

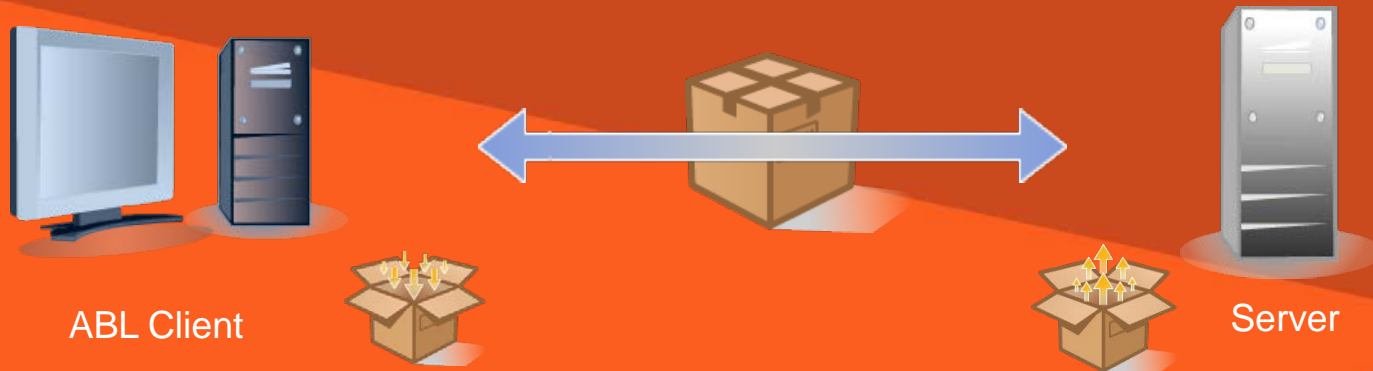
```
getCust.p at line 20 (c:\00\getCust.p)  
runit.p at line 2 (c:\00\runit.p)
```

Server StackTrace:

```
serverCust.p at line 8 (./serverCust.p)
```

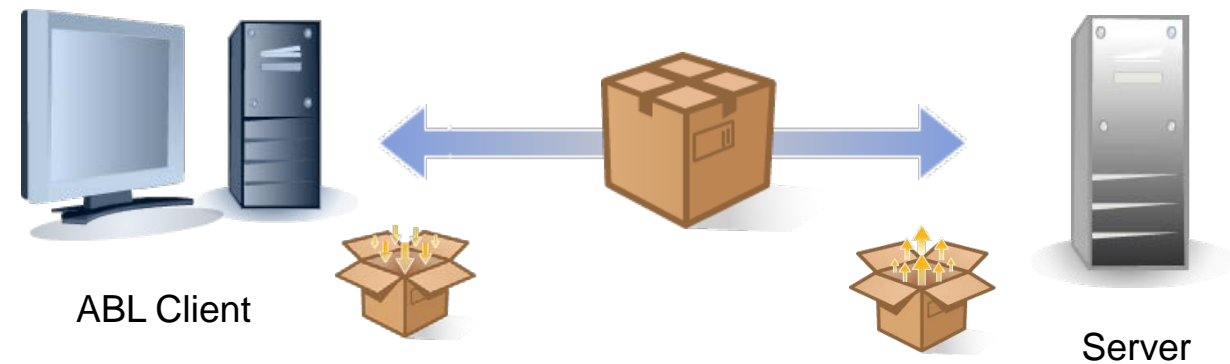
**RUN serverCust.p ON
SERVER hSrvr.**

Passing Objects between Client and Appserver



OO ABL Serialization

- How objects get passed between a client and an AppServer
- What objects can be serialized?
- Compatibility between client & server
- Serialization rules
- Deserialization rules



Passing OOABL Objects

- **Parameters**

```
RUN proc.p ON SERVER hsrv (INPUT myCustInfo).
```

- **Return Values**

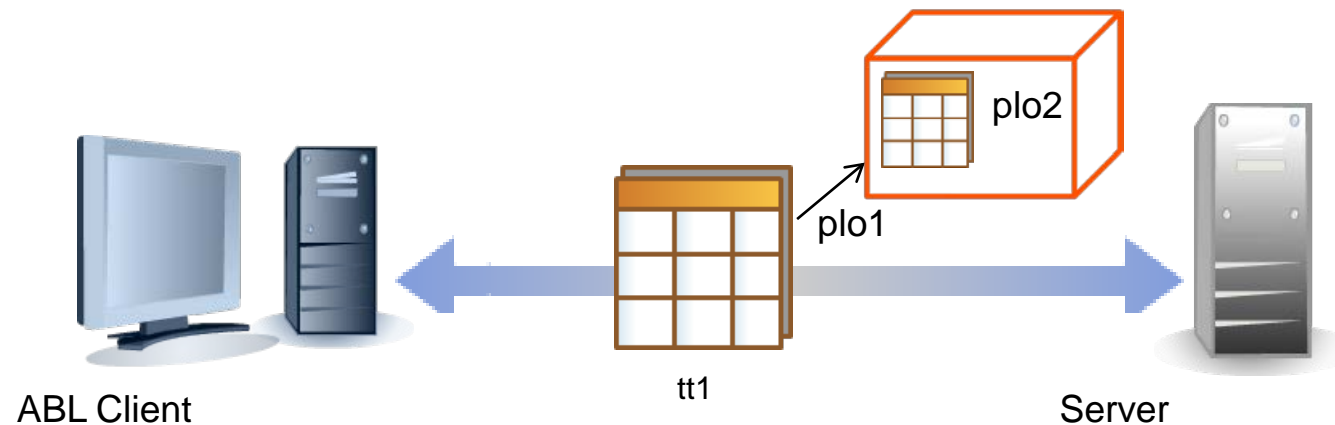
```
DEFINE VAR myCustInfo AS CustInfo.  
  
FUNCTION getData RETURNS CustInfo () IN hRemoteProc.  
...  
RUN CustServices.p ON SERVER hsrv SET hRemoteProc.  
...  
  
myCustInfo = getData().
```

Passing Remote Temp-tables Containing ABL Object Fields

■ Restriction lifted

- Pass temp-table to AppServer if it contains an OOABL object
- Field is still defined as `Progress.Lang.Object`

■ TT can contain object instance, which can contain TT...



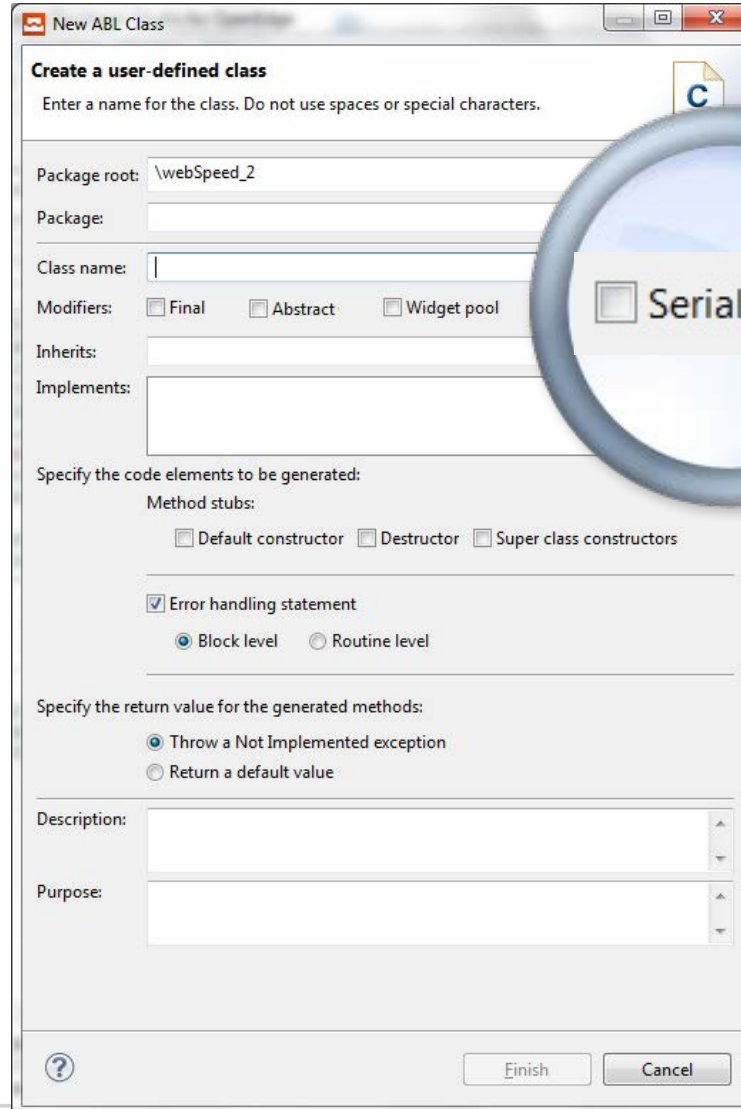
SERIALIZABLE

- Indicates objects of the class can be passed between an AppServer and a remote client

```
CLASS CustInfo INHERITS Info SERIALIZABLE:  
    ...  
END.
```

- Every class in hierarchy must be marked **SERIALIZABLE**
- Cannot be used with ABL-extended .NET classes

SERIALIZABLE – PDSOE



The image shows a screenshot of the 'New ABL Class' dialog box in a software development environment. The dialog is titled 'New ABL Class' and contains the following fields and options:

- Create a user-defined class**
Enter a name for the class. Do not use spaces or special characters.
- Package root: \webSpeed_2
- Package: (empty)
- Class name: (empty)
- Modifiers: Final Abstract Widget pool
- Inherits: (empty)
- Implements: (empty)
- Specify the code elements to be generated:
 - Method stubs:
 - Default constructor
 - Destructor
 - Super class constructors
 - Error handling statement
 - Block level
 - Routine level
- Specify the return value for the generated methods:
 - Throw a Not Implemented exception
 - Return a default value
- Description: (empty text area)
- Purpose: (empty text area)

At the bottom of the dialog are buttons for '?', 'Finish', and 'Cancel'. A magnifying glass is positioned over the 'Serializable' checkbox, which is currently unchecked.

Serializable Built-in OOABL Objects

■ **Serializable**

- Classes that implement `Progress.Lang.Error`
- `Progress.Json.ObjectModel.JsonObject`
- `Progress.Json.ObjectModel.JsonArray`
- `Progress.Json.ObjectModel.ObjectModelParser`
 - Any **built-in** sub-class of any of these
- `Progress.Lang.Object`

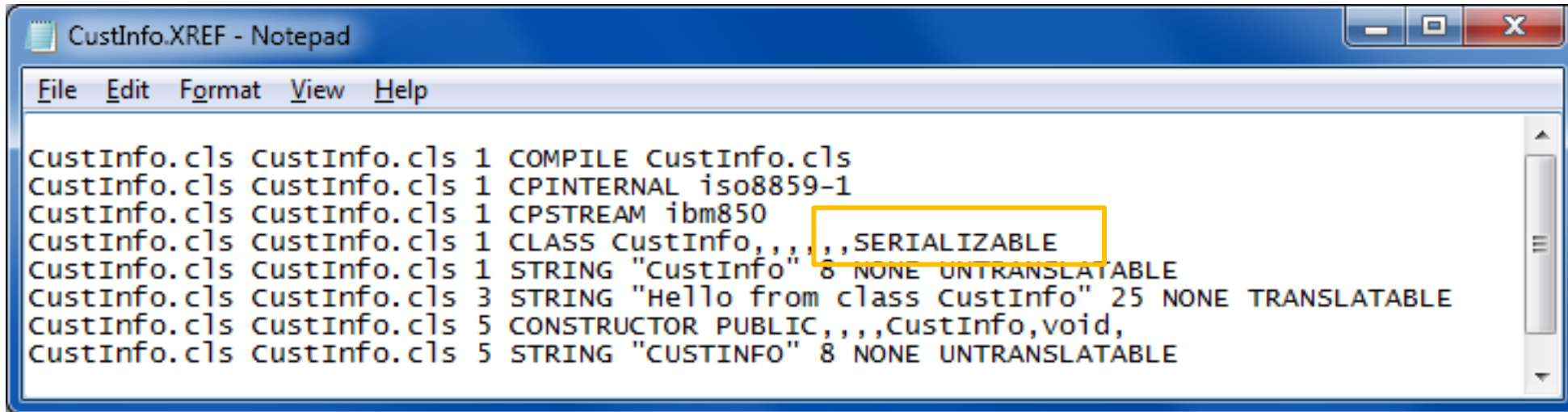
■ **Not serializable** – everything else, for example:

- `Progress.Security.DB.Policy`
- `Progress.Database.TempTableInfo`
- `Progress.BPM.DataSlot`
- `Progress.Lang.Class`

- **IsSerializable** method of **Progress.Lang.Class**
 - Indicates whether the object is SERIALIZABLE
 - Use at run-time or for tooling

```
DEFINE VAR cls AS Progress.Lang.Class  
cls = Progress.Lang.Class:GetClass("CustInfo").  
MESSAGE cls:IsSerializable() VIEW-AS ALERT-BOX.
```


Update to COMPILE XREF, COMPILE XREF-XML



```
CustInfo.XREF - Notepad
File Edit Format View Help
CustInfo.cls CustInfo.cls 1 COMPILE CustInfo.cls
CustInfo.cls CustInfo.cls 1 CPINTERNAL iso8859-1
CustInfo.cls CustInfo.cls 1 CPSTREAM ibm850
CustInfo.cls CustInfo.cls 1 CLASS CustInfo,,,,,SERIALIZABLE
CustInfo.cls CustInfo.cls 1 STRING "CustInfo" 8 NONE UNTRANSLATABLE
CustInfo.cls CustInfo.cls 3 STRING "Hello from class CustInfo" 25 NONE TRANSLATABLE
CustInfo.cls CustInfo.cls 5 CONSTRUCTOR PUBLIC,,,,CustInfo,void,
CustInfo.cls CustInfo.cls 5 STRING "CUSTINFO" 8 NONE UNTRANSLATABLE
```

<Class-ref>

<Source-guid>t6BMga8eOYXVE8DcTJMLng</Source-guid>

<Ref-seq>4</Ref-seq>

<Inherited-list/>

...

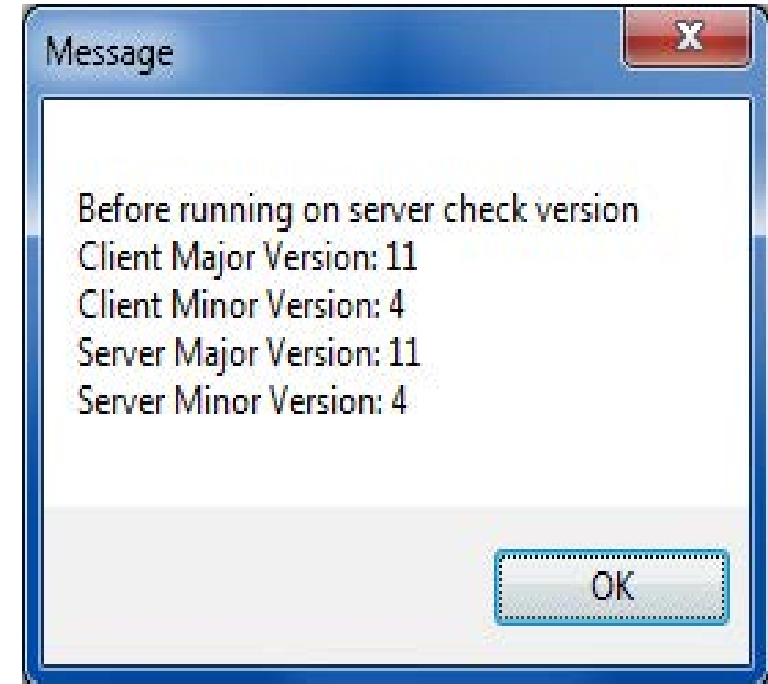
<Is-final>>true</Is-final>

<Is-serializable>>true</Is-serializable>

- **Both sides must be at least 11.4**
 - 11.4 client -> older AppServer
 - Parameter passing errors
 - 11.4 AppServer -> older client
 - OOABL error object not thrown
 - Parameter passing errors

Version Compatibility – VersionInfo Class

```
DEFINE VARIABLE hServer AS HANDLE.  
DEFINE VARIABLE clientVersionInfo AS Progress.Lang.OEVersionInfo.  
DEFINE VARIABLE serverVersionInfo AS Progress.Lang.OEVersionInfo.  
  
CREATE SERVER hServer.  
hServer:CONNECT ("-AppService asbroker1 -H localhost -S 3090") NO-ERROR.  
  
IF hServer:CONNECTED () THEN  
DO:  
    clientVersionInfo = hServer:REQUEST-INFO:VersionInfo.  
    serverVersionInfo = hServer:RESPONSE-INFO:VersionInfo.  
  
    MESSAGE "Before running on server check version" SKIP  
        "Client Major Version:" clientVersionInfo:OEMajorVersion SKIP  
        "Client Minor Version:" clientVersionInfo:OEMinorVersion SKIP  
        "Server Major Version:" serverVersionInfo:OEMajorVersion SKIP  
        "Server Minor Version:" serverVersionInfo:OEMinorVersion SKIP  
    VIEW-AS ALERT-BOX.  
  
    IF serverVersionInfo:OEMinorVersion >= 4 THEN DO:  
        ...  
    END.  
END.
```



Serialization Model

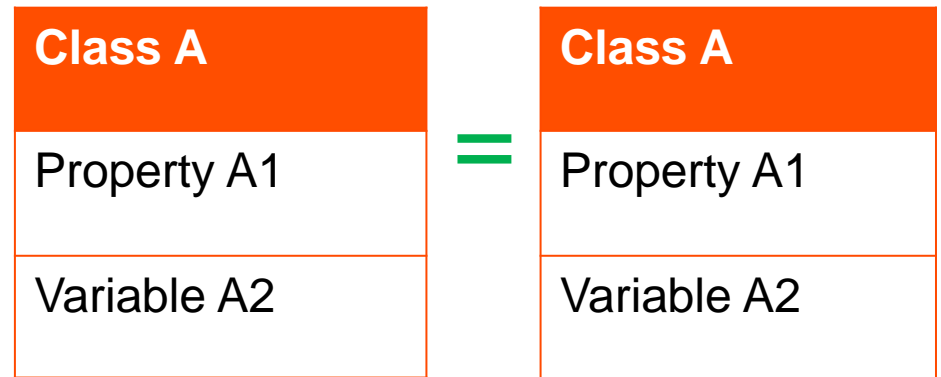
■ Pass by value

- Receiving side creates **new** object instance
- Either instance may get garbage collected



Compatibility: Class Definitions

- **Class definitions on Client and AppServer must be the “*same*”**
 - Method signature and data members must match *exactly*
- **What if they are different?**
 - An error is raised on the RUN statement
- **AVM does not check if the business logic matches**
 - Constructor, method or property getter/setter code can be different
 - API the same, r-code is different



What Gets Serialized?

- **All instance data members are serialized**
 - Variables
 - Properties
 - ProDataSets
 - Temp-tables
- **All access modes**
 - Public, Protected, Private
- **Static** data members are **NOT** serialized
- **Property getters**
 - **Not** invoked
 - Value is copied



Serialization Rules – Special Cases

■ MEMPTRs

- Serialize if allocated by the ABL application
- Not serialized if allocated from external sources
 - DLL or shared library
 - Set to Unknown when the object is deserialized

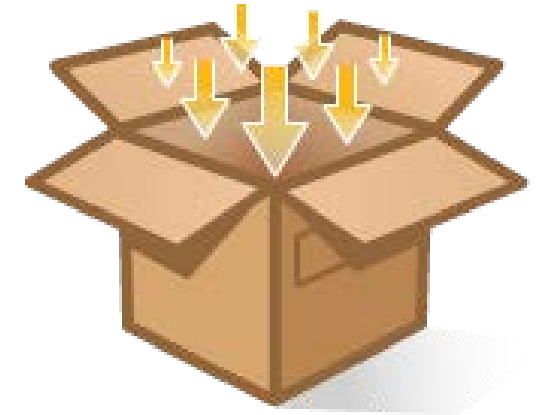
■ Handle-based variables (e.g., widgets, queries, buffers)

- Serialized with the handle value
- Widget/object referenced by the handle is not serialized
- Only useful to round-trip data

■ Cannot serialize .NET or ABL-extended .NET objects

- AVM raises an error

**Handle
with Care**



Serialization Rules – State

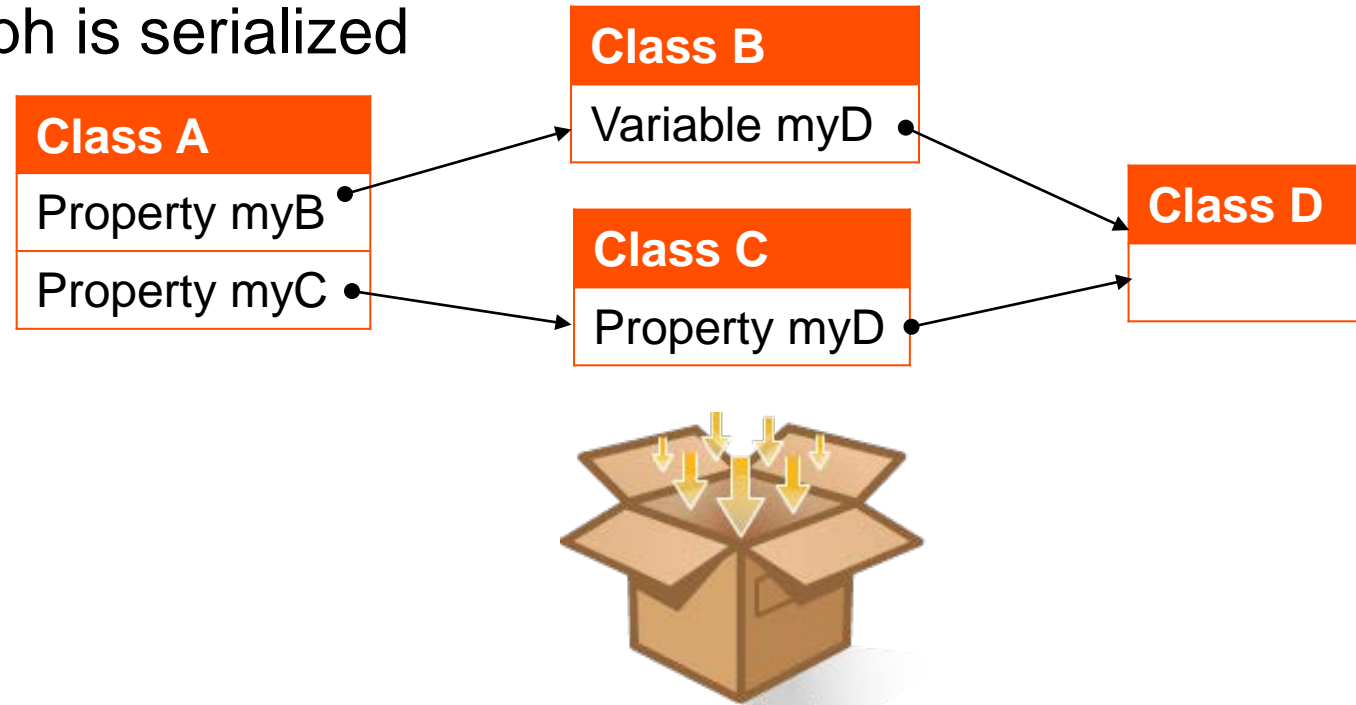
- The AVM does not maintain state of class instance
 - Open queries/cursor position
 - Buffer contents
 - Open files
 - Streams
 - Event subscriptions



Serialization Rules – Object Relationships

- Deep-copy

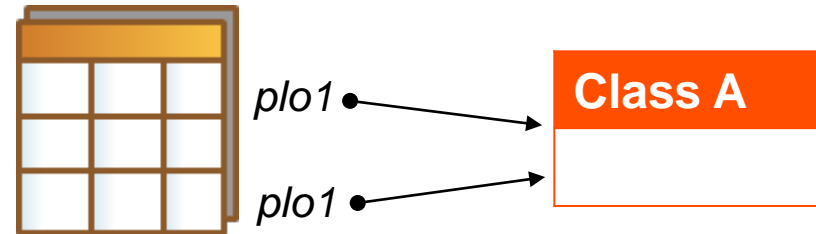
- Serialize data member object references
- Object graph is serialized



- Only 1 instance of Class D is serialized

Temp-Tables and Object Fields

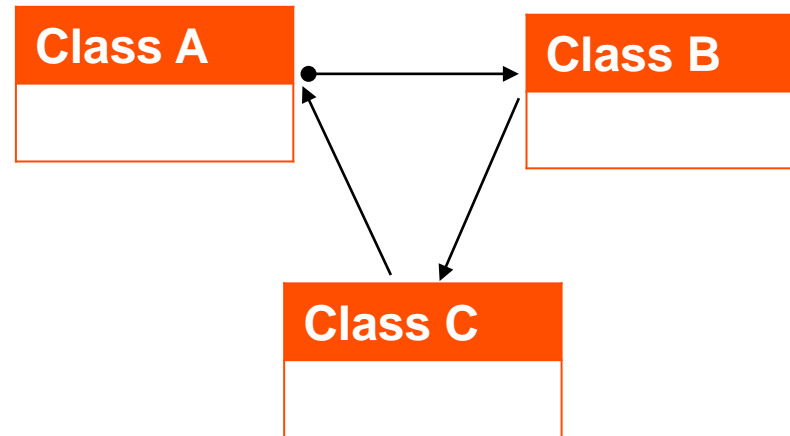
- Multiple references to one instance
 - Instance uniqueness is maintained



- Only 1 instance of Class A is serialized

Circular References

- Circular references are detected and **OK**
 - No infinite loop 😊



Deserialization Rules

- Creating the new instance
 - Instance Constructor **not** invoked
 - Property Setters **not** invoked
- Only the object's data is deserialized
 - R-code must already exist on both sides of the wire



DynObjects Logging

- DynObjects logging includes objects created by deserialization
- Use LOG-ENTRY-TYPES: DynObjects.Class

```
RUN objParm.p ON hServer (INPUT NEW classA()).
```

```
[14/07/21 @ 13:56:26.322-0400] P-008364 T-009896 3 AS DYNOBJECTS    Created  
Progress.Lang.Object   Handle:1000 (objParm.p @ 0) classA
```

Serialization of Character Data

- Character data serialized via sender's `-cpinternal`
- Character data deserialized via receiver's `-cpinternal`
- Longchar same rules apply except if:
 - Codepage fixed with `FIX-CODEPAGE`



ABL Client



Server

Character -cpinternal

Character -cpinternal

- Runtime error can be raised during conversion

Object Serialization – On the Roadmap

- Transient data (do not serialize)
- Provide object serialization to disk
 - Binary format
 - JSON
 - XML

```
DEFINE PUBLIC VARIABLE x AS INT.  
DEFINE PUBLIC VARIABLE y AS INT.  
DEFINE PUBLIC VARIABLE z AS INT.
```

- Provide options to support “relaxed” levels of client/server matching:
 - Exact match for public and protected members only
 - Match by data members name & type
- Application defined (via callback)

```
DEFINE PUBLIC VARIABLE y AS INT.  
DEFINE PUBLIC VARIABLE x AS INT.  
DEFINE PUBLIC VARIABLE w AS INT.
```

Agenda

- OOABL serialization
- FINALLY block
- GET-CLASS
- JSON Before-Image Support
- 64-bit WebClient

FINALLY Block – Motivation

Problem

Flow-of-control statements in a FINALLY block may conflict with associated block

```
DO TRANSACTION:  
    UNDO THROW myAppError.  
END.  
  
FINALLY:  
    RETURN.  
END.
```

Solution

We changed how the AVM handles flow-of-control statements in a FINALLY block

FINALLY Block

Associated Block	FINALLY block	Caller
Return 1	Return 2	2
Error 1	RETURN, NEXT, LEAVE, RETRY	Error 1

- 2nd line is new behavior in 11.4
- Best Practice: Avoid flow-of-control conflicts between Associated block and FINALLY block

Agenda

- OOABL serialization
- FINALLY block
- GET-CLASS
- JSON Before-Image Support
- 64-bit WebClient

GET-CLASS – Motivation

Problem – Prior to 11.4

- ABL supports Progress.Lang.Class:GetClass(<type-name-exp>)
- This does not provide *compile time* validation of type-name-exp

Solution

- Introduce GET-CLASS built-in function
- Accepts a type-name parameter
 - not a character expression

GET-CLASS

- Syntax

```
GET-CLASS(<type-name>).
```

- Returns a Progress.Lang.Class
- USING statements are applied to a non-qualified name
- Compiler error if not found

Agenda

- OOABL serialization
- FINALLY block
- GET-CLASS
- JSON Before-Image Support
- 64-bit WebClient

JSON – Before-Image – Motivation

Problem

- Lack of serialize / deserialize for a ProDataSet with before-image data to JSON
- Out of step with XML support
- **Mobile**

Solution

- Optional before-image data in JSON for ProDataSets

JSON – Before-Image – Motivation

- **You cannot reliably save ProDataSet changes to the DB w/o a before-image**
 - You cannot know if another user has changed the data first.
- **Mobile**
 - Original version – “built-in” method for update only handled 1 record at a time.
 - Application would have to do its own before-image caching and checking
 - In 11.4 – Added ability to return a **set** of records in a ProDataSet.
 - Requires reliable SAVE-ROW-CHANGES – need Before-image Information
- **Offline support**
 - Make updates to a ProDataSet; Save to JSON since DB is unavailable
 - Read back later when connected and do SAVE-ROW-CHANGES

JSON – Before-Image Syntax

- Syntax:

```
WRITE-JSON ( target-type , { file | stream | stream-handle | memptr | longchar }  
            [ , formatted [ , encoding [ , omit-initial-values  
            [ , omit-outer-object [ , write-before-image ] ] ] ] )
```

- Example:

```
DEFINE VARIABLE writeBI AS LOGICAL INIT YES.  
DATASET dset:WRITE-JSON ( "File", "test.json", YES, "UTF-8", YES,  
                          NO, YES).
```

- No change to READ-JSON syntax

ProDataSet – JSON Output

After table (current state)

```
{ "dsCustomer": {  
  "prods:hasChanges": true,  
  "ttCust": [  
    {  
      "prods:id": "ttCust10497",  
      "prods:rowState": "modified",  
      "CustNum": 2,  
      "NAME": "Urpon Frisbee_NewName",  
      "Balance": 903.64  
    },  
  ],  
}
```

Before table

```
"prods:before": {  
  "ttCust": [  
    {  
      "prods:id": "ttCust10497",  
      "prods:rowState": "modified",  
      "CustNum": 2,  
      "NAME": "Urpon Frisbee",  
      "Balance": 437.63  
    },  
  ],  
}
```

**Record marked as
"modified"**

ProDataSet – Before Table May Also Indicate Row Error

```
...
"prods:before": {
  "ttCust": [
    {
      "prods:id": "ttCust10520",
      "prods:rowState": "deleted",
      "prods:hasErrors": true,
      "CustNum": 3,
      "NAME": "Hoops",
      "Balance": 1199.95
    },
    ...
  ],
  "prods:errors": {
    "ttCust": [
      {
        "prods:id": "ttCust10520",
        "prods:error": "error-string"
      },
      ...
    ]
  }
}
```

*Error associated with
this row*

*If row not deleted,
hasErrors would be
in after table instead*

Agenda

- OOABL serialization
- FINALLY block
- GET-CLASS
- JSON Before-Image Support
- 64-bit WebClient

WebClient – Windows 64-bit

Problem – Since 11.3

- Provided a 64-bit GUI client
- Missing functionality – no support for 64-bit WebClient

Solution

WebClient application can be defined as supporting

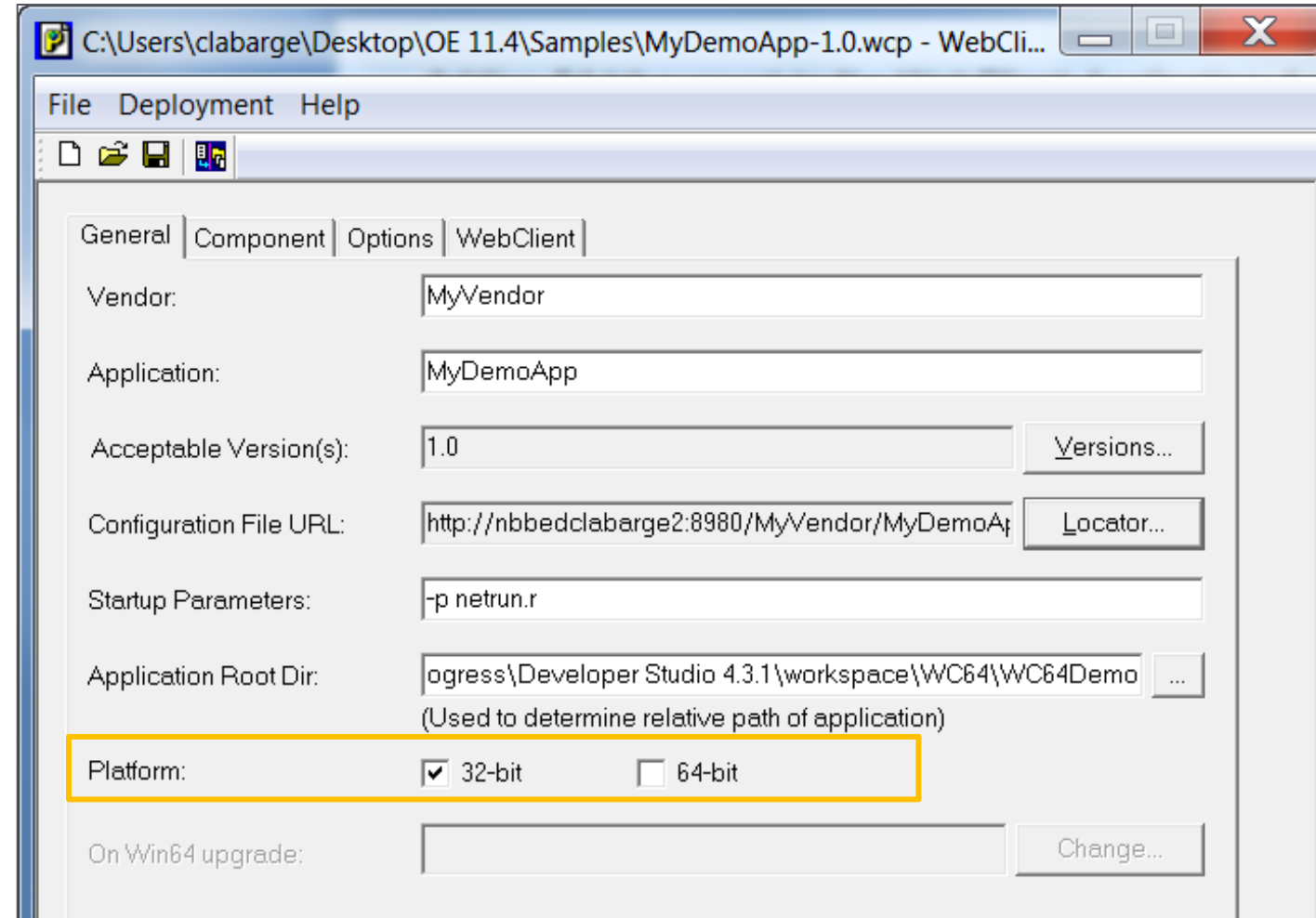
- 32-bit platform
- 64-bit platform
- Either, depending on target machine

WebClient – Windows 64-bit

- **When your application gets deployed**
 - WebClient (i.e., the Progress AVM) is installed if not already there
 - The app gets installed
 - In general ABL code is not impacted by 32-bit vs. 64-bit
 - If it is, it can/should be conditionalized to support both versions
 - But the install is targeted for **either** 32-bit or 64-bit
 - Notably – we need to know which AVM to run
- We support both 32-bit and 64-bit WebClient on the same machine
 - 2 different applications, one 32-bit, one 64-bit
 - Do **NOT** support this for the GUI client in 11.4

WebClient Application Assembler – General Tab

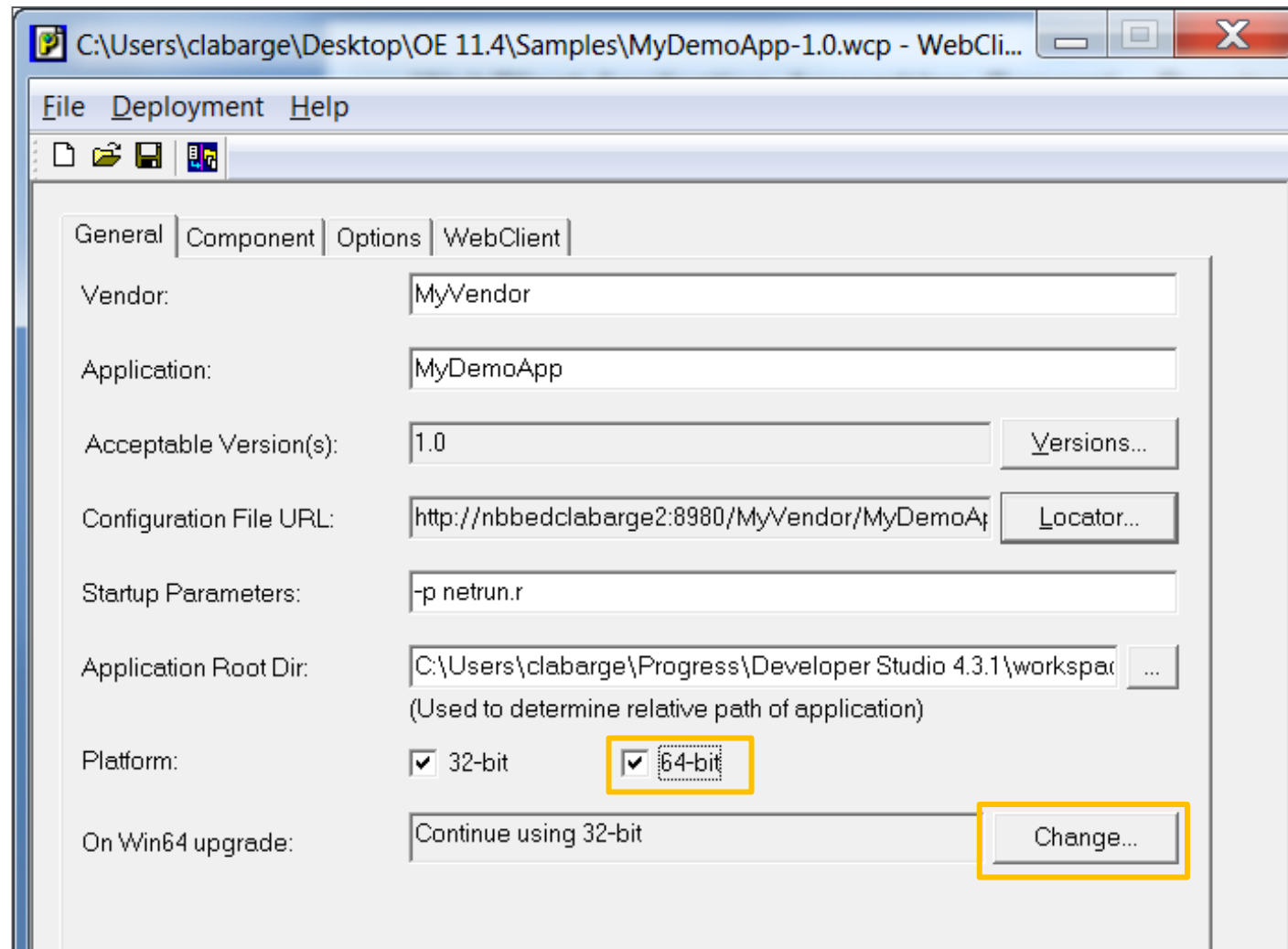
- On **General** tab, added
 - Platform toggles
 - 32-bit
 - 64-bit
 - Pick one or the other
 - Pick both:
 - Install will match the machine configuration
 - Will end up with 32-bit **and** 64-bit AVM if:
 - 64-bit machine
 - Another 32-bit app already installed
 - Your app is installed as 64-bit



WebClient Application Assembler – Application Upgrade

- When you select both 32-bit & 64-bit
- You, the developer, decide the **upgrade** path:

- Continue to run the application as 32-bit
- Uninstall 32-bit version and install 64-bit version
- Ask the end-user: keep 32-bit or upgrade to 64-bit



Want to Learn More About OpenEdge 11?

- Role-based learning paths are available for OpenEdge 11
- Each course is available as Instructor-led training or eLearning
- Instructor-led training:
 - \$500 per student per day
 - www.progress.com/support-and-services/education/instructor-led-training
- eLearning:
 - Via the Progress Education Community (wbt.progress.com):
 - OpenEdge Developer Catalog: \$1500 per user per year
 - OpenEdge Administrator Catalog: \$900 per user per year
- User Assistance videos:
www.progress.com/products/pacific/help/openedge

PROGRESS EXCHANGE²⁰¹⁴

Visit the Resource Portal

- **Get session details & presentation downloads**
- **Complete a survey**
- **Access the latest Progress product literature**

www.progress.com/exchange2014



PROGRESS