What's New in the ABL in Progress OpenEdge Release 11

Matt Gilarde – Principal Software Engineer @ Progress
Cindy Winer – Principal QA Engineer @ Progress

October 7, 2013

PROGRESS EXCHANGE 2013
DISCOVER. DEVELOP. DELIVER.

What’s New in the ABL in Progress OpenEdge Release 11

- Upgrades
- Security
- New Language Features
- Windows 64-bit GUI Client
- Questions & Answers

What’s New in the ABL in Progress OpenEdge Release 11

- **Upgrades**
- **Security**
- New Language Features
- Windows 64-bit GUI Client
- Questions & Answers

Upgrades

- Xerces XML Parser was upgraded from IBM 5.2.0 to Apache 3.1.1 (Release 11.1)
  - Used by the ABL’s XML features
  - Apache Xerces 3.1.1 is the current version of Apache Xerces
  - Apache Xerces is based on IBM Xerces
- Upgraded Java Service Data Object (SDO) API from 1.0 to 2.1.1 (Release 11.2)
  - Used by the Java Open Client’s ProDataGraph interface
  - SDO 2.1 is the current standard for Java SDOs (JSR 235)
- Both upgrades provide bug fixes and better performance

ABL-based User Authentication (Release 11.1)

Problem:

I need to perform custom user authentication.

Solution:

You can now write custom authentication code in ABL.

- Works with the OpenEdge Identity Management (IdM) Framework
- Authentication can be configured to use an ABL plug-in
- During user authentication the ABL plug-in calls the Authenticate-User ABL callback procedure
- Allows decoupling of authentication from the application

Encoded Password (Release 11.1)

Problem:

I have to use clear-text passwords in scripts and other visible places.

Solution:

You can now use encoded passwords so they are no longer readable to users.

- Use encoded passwords in place of clear-text passwords
- Encoded passwords may be used with –P, SETUSERID(), and Client-principal objects
- Generate encoded passwords with genpassword.exe or the AUDIT-POLICY:ENCRYPT-AUDIT-MAC-KEY() method

What’s New in the ABL in Progress OpenEdge Release 11

- Upgrades
- Security
- **New Language Features**
- Windows 64-bit GUI Client
- Questions & Answers

LIKE Phrase for Functions and Method Parameters (Release 11.1)

Problem:

- I want to use the LIKE phrase to define parameters for methods and functions so I can keep the parameters consistent throughout my application.

Solution:

- LIKE phrase can now be used to define parameters for methods and functions

```
METHOD PUBLIC VOID showLikeParms
       (INPUT cName LIKE Customer.Name, /* Database   */
        INPUT eName LIKE ttEmail.Name,  /* Temp-table */
        INPUT tVar  LIKE testVar):      /* Variable   */
END.
```

Sub-second PAUSE (Release 11.2)

Problem:

- I can not use a fractional value in the **`PAUSE`** statement, therefore I cannot pause for less than 1 second.

Solution:

- **`PAUSE`** now allows the AVM to process a fractional value of `n`

Before:	PAUSE 1
After:	PAUSE .5

ProVersion enhancement (Release 11.2)

Problem:

I need to know exactly which release my application is running on.

Solution:

PROVERSION() now provides service pack, hotfix, and build numbers.

- PROVERSION() returns major and minor release numbers (e.g.: 11.3)
- PROVERSION(1) plus service pack, hotfix, and build numbers (e.g.: 11.3.1.001.1234)

RCODE-INFO:DISPLAY-TYPE (Release 11.3)

Problem:

- I need to determine programmatically which display environment (GUI or TTY) this code must execute in.

Solution:

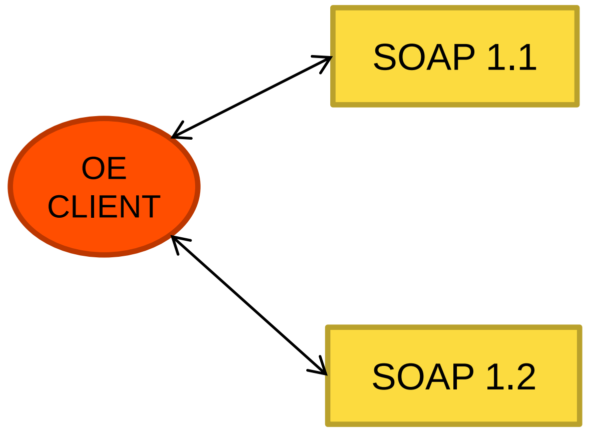
- New attribute on the **RCODE-INFO** system handle:
  - **DISPLAY-TYPE**

Return value	Has UI statements	COMPILE/RUN On
“GUI”	Yes	GUI
“TTY”	Yes	TTY
“” (empty string)	No	Any

SOAP 1.2 (Release 11.3)

Problem:

- I need to be able to access a SOAP 1.2 web service from the ABL client



Solution:

- The OE client now supports accessing a SOAP 1.2 web service.
- New attribute on server handle, hWebService:**SOAP-VERSION**
  - Returns "1.1" or "1.2" depending on the web service you connected to
- WSDL Analyzer : now includes the **SOAP VERSION** on the service.html page
  - SOAP 1.1 and/or SOAP 1.2
- What should I know to use this?

  There are new attributes and methods on the **SOAP-HEADER-ENTRYREF** and **SOAP-FAULT** objects

SOAP 1.2

- Changes to attributes and methods for **SOAP-HEADER-ENTRYREF**

SOAP 1.1	SOAP 1.2	
ACTOR	ROLE	The URI of the recipient of the SOAP header
SET-ACTOR()	SET-ROLE()	Set the URI of the recipient of the SOAP header

SOAP 1.2

- Four new attributes have been added to the **SOAP-FAULT** object handle, these attributes are only applicable for SOAP 1.2:

Attribute	Returns
SOAP-FAULT-NODE	returns the URI of the Web service node that caused the fault
SOAP-FAULT-ROLE	returns the URI that identifies the role the node was operating in at the point the fault occurred.
SOAP-FAULT-SUBCODE	returns a list of SOAP fault sub-code names for the fault
SOAP-FAULT-MISUNDERSTOOD-HEADER	returns a list of SOAP header names resulting from “MustUnderstand” faults.

SOAP 1.2

- CONNECTING to the web service

Hint

- WSDL has both SOAP 1.1 and SOAP 1.2 ports, the default connection is to SOAP 1.1

- Must provide more info when connecting to the SOAP 1.2 web service

Block Level Undo Throw Directive (Release 11.2)

Problem:

- Cannot change the default error directive on REPEAT, FOR, or DO TRANSACTION blocks

Solution:

- Added BLOCK-LEVEL error directive

- **BLOCK-LEVEL ON ERROR UNDO, THROW.** statement
  - Changes the default for all blocks in a file that have a default error directive, including routine blocks, to have the UNDO, THROW error directive instead
    - Routine-level blocks
    - **REPEAT** blocks
    - **FOR** blocks
    - **DO TRANSACTION** blocks

Block Level Undo Throw Directive

Problem:

- I don’t want to have to change all of the files in my application and I need to be able to tell which files are using which error directive.

Solution:

- New startup parameter: **-undothrow n**, used at compile time

  Behaves as if the statement was inserted in every source code file being compiled

  - n=1, **ROUTINE-LEVEL** statement
  - n=2, **BLOCK-LEVEL** statement

```
> prowin32.exe -p compile-application.p -undothrow 2
```

This will change the behavior of your application

- The app must be coded accordingly

Hint

Block Level Undo Throw Directive

- New attribute on the **RCODE-INFO** system handle:
  - **UNDO-THROW-SCOPE**   the error handling directive in effect
    - **"ROUTINE-LEVEL"**
    - **"BLOCK-LEVEL"**
    - “”

- New line in the output of **COMPILE XREF/ XREF-XML** indicates whether **`ROUTINE-LEVEL`** or **`BLOCK-LEVEL`** was specified at compile time

- Examples

classname.cls classname.cls 1 BLOCK-LEVEL ON ERROR UNDO, THROW

<Reference Reference-type="ROUTINE-LEVEL" Object-identifier="">

Dynamic Access to Built-in Objects (Release 11.3)

Problem:

- I want to be able to call ABL built-in OO objects dynamically.

Solution:

- The ABL built-in OO objects can now be accessed dynamically, using either Dynamic-* functions or Reflection with the Progress.Lang.Class methods.

- This applies to the following built-in objects:
  - `Progress.BPM.*`
  - `Progress.Data.*`
  - `Progress.Json.*`
  - `Progress.Security.*`
  - `Progress.Lang.*`

Dynamic Access to Built-in Objects

- Non Dynamic

```
DEFINE VARIABLE jaCustomer AS Progress.Json.ObjectModel.JsonArray.
jaCustomer = NEW Progress.Json.ObjectModel.JsonArray().
```

- Dynamic

```
DEFINE VARIABLE jaCustomer AS Progress.Json.ObjectModel.JsonArray.
jaCustomer = DYNAMIC-NEW “Progress.Json.ObjectModel.JsonArray”().
```

- Dynamic with Reflection

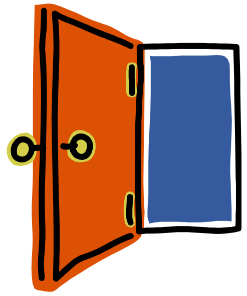
```
DEFINE VARIABLE plcjArray AS Progress.Lang.Class.
plcjArray =
Progress.Lang.Class:GetClass("Progress.Json.ObjectModel.JsonArray").
myPLO = plcjArray:New().
```

Dynamic Access to Built-in Objects

- Dynamic functions
  - DYNAMIC-NEW()
  - DYNAMIC-INVOKE()
  - DYNAMIC-PROPERTY()

- Reflection using methods on Progress.Lang.Class
  - plcobj:New()
  - plcobj:Invoke()
  - plcobj:GetPropertyValue()
  - plcobj:SetPropertyValue()

Class Private and Protected Data Members (Release 11.3)



Problem:

- The ABL doesn't work like other OO languages when it comes to accessing Private and Protected data members.

Solution:

- Private and protected class members have changed from **instance** based to **class** based access.
- Instance-based (then)

  Private/protected class members could only be accessed in the current instance of the class / hierarchy

- Class-based (now)

  Private/protected data members can be accessed from another instance of the same class / hierarchy

- This change affects variables, properties, methods and events

Class Private and Protected Data Members

Class-based access

- Private class members

  The object reference must be the <u>same class</u> as the one you are executing in.

- Protected class members:

  The object reference must be <u>type compatible</u> with the class you are executing in.

Class Private and Protected Data Members

- Private data member example

```
CLASS class1:

      DEFINE PRIVATE VARIABLE myPrivVar AS CHARACTER.


      METHOD PUBLIC VOID mRunit():
            /* Assign to a private variable in this instance of the class  */
            myPrivVar = “inside my instance of the class”.


           DEFINE VARIABLE newInstance AS class1.
           newInstance = NEW class1().
           /* Assign to a private variable in another  instance of the same class  */
          newInstance:myPrivVar = “some other text in new instance".
   END.
END.
```

Class Private and Protected Data Members

- Private data member example, cont.

```
CLASS class1:

    DEFINE PRIVATE VARIABLE myPrivVar AS CHARACTER.


    METHOD PUBLIC VOID mRunit():
        /* Assign to a private variable in this instance of the class  */
        myPrivVar = “inside my instance of the class”.


        DEFINE VARIABLE newInstance AS class1.
        newInstance = NEW class1().
        /* Assign to a private variable in another  instance of the same class  */
        newInstance:myPrivVar = “some other text in new instance".
  END.
END.
```

Class Private and Protected Data Members

- Private data member example, cont.

```
CLASS class1:

      DEFINE PRIVATE VARIABLE myPrivVar AS CHARACTER.


      METHOD PUBLIC VOID mRunit():
            /* Assign to a private variable in this instance of the class  */
            myPrivVar = “inside my instance of the class”.


           DEFINE VARIABLE newInstance AS class1.
           newInstance = NEW class1().
           /* Assign to a private variable in another  instance of the same class  */
          newInstance:myPrivVar = “some other text in new instance".
   END.
END.
```

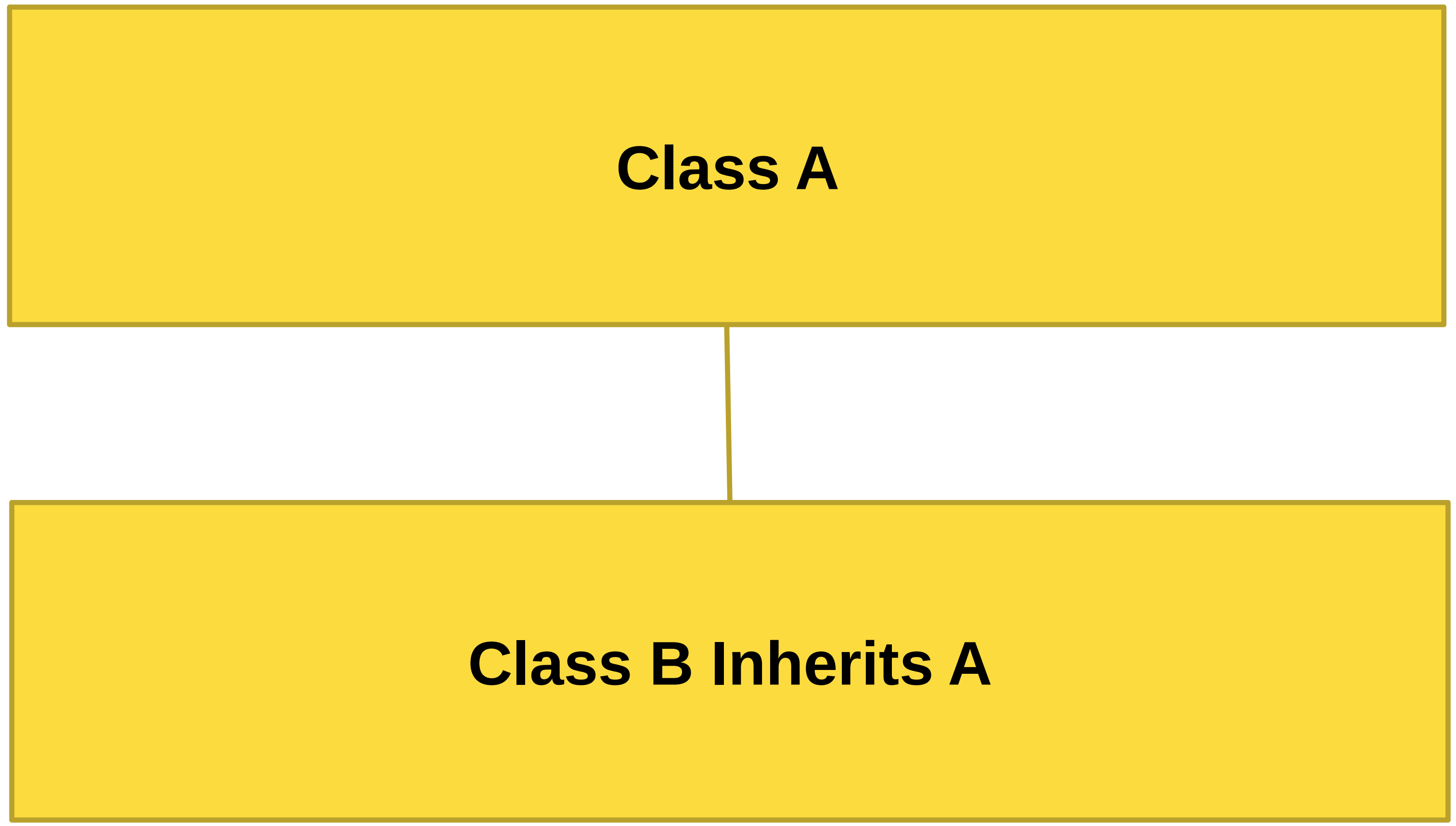
Class Private and Protected Data Members

- Private data member example, cont.

```
CLASS class1:

      DEFINE PRIVATE VARIABLE myPrivVar AS CHARACTER.


      METHOD PUBLIC VOID mRunit():
            /* Assign to a private variable in this instance of the class  */
            myPrivVar = “inside my instance of the class”.


           DEFINE VARIABLE newInstance AS class1.
           newInstance = NEW class1().
           /* Assign to a private variable in another  instance of the same class  */
          newInstance:myPrivVar = “text in new instance".
   END.
END.
```

Class Private and Protected Data Members

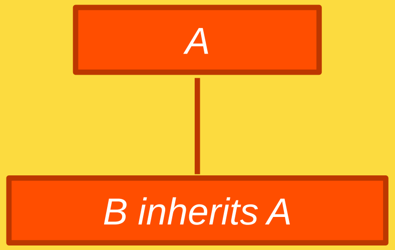
- Hierarchy for Protected example



Class Private and Protected Data Members

- Protected data member example, cont.

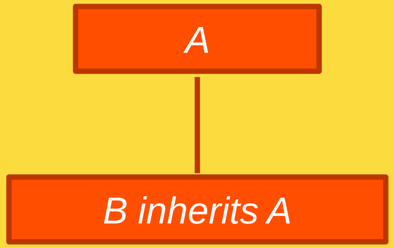
```
CLASS A:
  DEFINE PROTECTED VARIABLE myprotVar  AS CHARACTER.
  METHOD PUBLIC VOID runme():
    DEFINE VARIABLE aobj AS CLASS A.
    DEFINE VARIABLE bobj AS CLASS B.
    aobj = NEW A().
    bobj = NEW B().
    aobj:myprotVar = "hello".
    bobj:myprotVar = "goodbye".
  END.
END.
```



Class Private and Protected Data Members

- Protected data member example, cont.

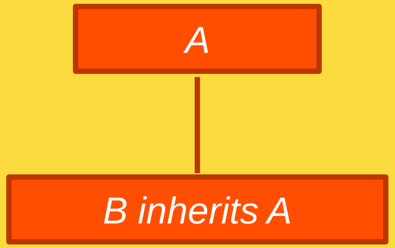
```
CLASS A:
  DEFINE PROTECTED VARIABLE myprotVar  AS CHARACTER.
  METHOD PUBLIC VOID runme():
    DEFINE VARIABLE aobj AS CLASS A.
    DEFINE VARIABLE bobj AS CLASS B.
    aobj = NEW A().
    bobj = NEW B().
    aobj:myprotVar = "hello".
    bobj:myprotVar = "goodbye".
  END.
END.
```



Class Private and Protected Data Members

- Protected data member example, cont.

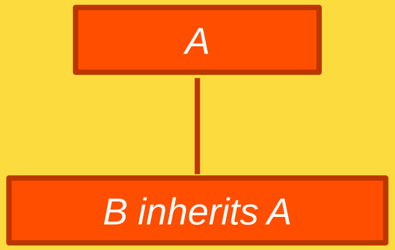
```
CLASS A:
  DEFINE PROTECTED VARIABLE myprotVar  AS CHARACTER.
  METHOD PUBLIC VOID runme():
    DEFINE VARIABLE aobj AS CLASS A.
    DEFINE VARIABLE bobj AS CLASS B.
    aobj = NEW A().
    bobj = NEW B().
    aobj:myprotVar = "hello".
    bobj:myprotVar = "goodbye".
  END.
END.
```



Class Private and Protected Data Members

- Protected data member example, cont.

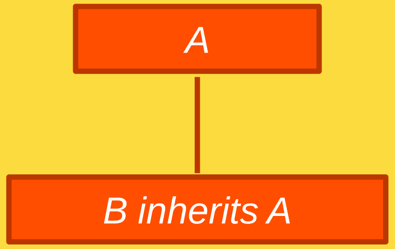
```
CLASS A:
   DEFINE PROTECTED VARIABLE myprotVar  AS CHARACTER.
   METHOD PUBLIC VOID runme():
      DEFINE VARIABLE aobj AS CLASS A.
      DEFINE VARIABLE bobj AS CLASS B.
      aobj = NEW A().
      bobj = NEW B().
      aobj:myprotVar = "hello".
      bobj:myprotVar = "goodbye".
   END.
END.
```



Class Private and Protected Data Members

- Protected data member example, cont.

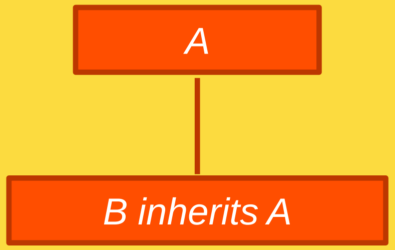
```
CLASS B INHERITS A:
   METHOD PUBLIC VOID runme2():
      DEFINE VARIABLE aobj AS CLASS A.
      DEFINE VARIABLE bobj AS CLASS B.
      aobj = NEW A().
      bobj = NEW B().
      bobj:myprotVar = “au revior".
      /* compile error: aobj is not type compatible with B */
      /* aobj:myprotVar = “bonjour".   */
  END.
END.
```



Class Private and Protected Data Members

- Protected data member example, cont.

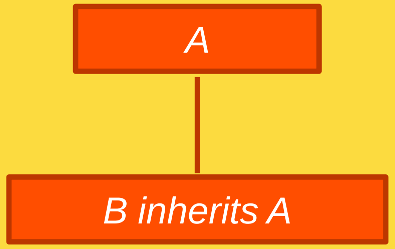
```
CLASS B INHERITS A:
    METHOD PUBLIC VOID runme2():
        DEFINE VARIABLE aobj AS CLASS A.
        DEFINE VARIABLE bobj AS CLASS B.
        aobj = NEW A().
        bobj = NEW B().
        bobj:myprotVar = “au revior".
        /* compile error: aobj is not type compatible with B */
        /* aobj:myprotVar = “bonjour".   */
   END.
END.
```



Class Private and Protected Data Members

- Protected data member example, cont.

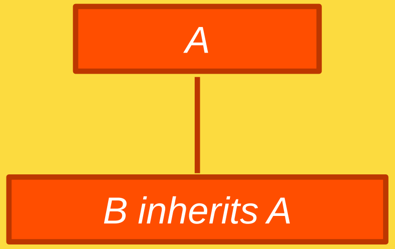
```
CLASS B INHERITS A:
    METHOD PUBLIC VOID runme2():
        DEFINE VARIABLE aobj AS CLASS A.
        DEFINE VARIABLE bobj AS CLASS B.
        aobj = NEW A().
        bobj = NEW B().
        bobj:myprotVar = “au revior".
        /* compile error: aobj is not type compatible with B */
        /* aobj:myprotVar = “bonjour".   */
   END.
END.
```



Class Private and Protected Data Members

- Protected data member example, cont.

```
CLASS B INHERITS A:
   METHOD PUBLIC VOID runme2():
      DEFINE VARIABLE aobj AS CLASS A.
      DEFINE VARIABLE bobj AS CLASS B.
      aobj = NEW A().
      bobj = NEW B().
      bobj:myprotVar = “au revior".
      /* compile error: aobj is not type compatible with B */
      /* aobj:myprotVar = “bonjour".   */
  END.
END.
```



Support for Single-Run / Singleton (Release 11.2/11.3)

Problem 1:

- When running a remote internal procedure on the AppServer, at least 3 trips between client and AppServer are necessary:
  1. Establish the persistent procedure and execute the main block
  2. Run an internal procedure within the persistent procedure
  3. Delete the persistent procedure

Problem 2:

- An AppServer agent is bound (or dedicated) to a particular client

Solution:

- Introduced support for Single-Run / Singleton

Support for Single-Run / Singleton

- Benefits
  - Reduce trips between client and AppServer for increased performance
  - Eliminates an AppServer agent from getting bound to a particular client
  - Client requests can go to any AppServer agent, no context information is retained

- Available with stateless and state-free AppServers

Hint

- A .p that is used for Single-Run/Singleton cannot have parameters on the main block.

Support for Single-Run / Singleton - Persistent model

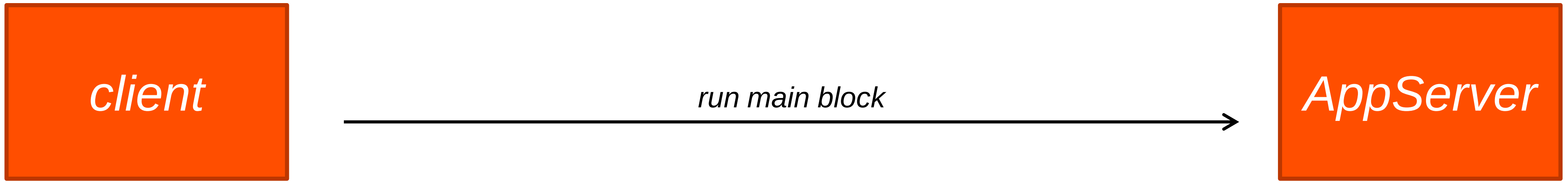
RUN server1.p ON hServer **PERSISTENT** SET hProc.

RUN internalproc IN hProc .

DELETE PROCEDURE hProc.

Support for Single-Run / Singleton - Persistent model

RUN server1.p ON hServer **PERSISTENT** SET hProc.

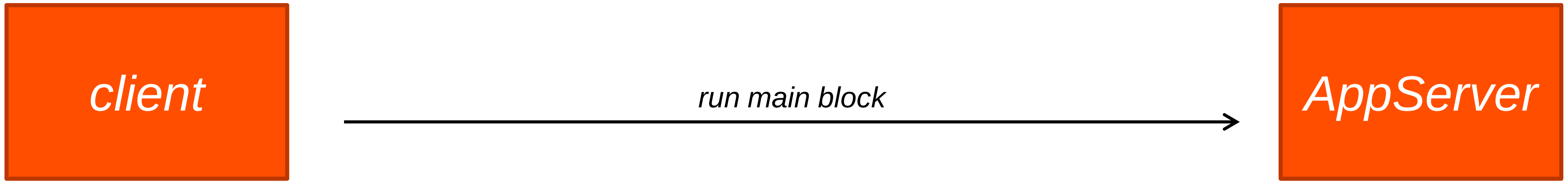


RUN internalproc IN hProc .

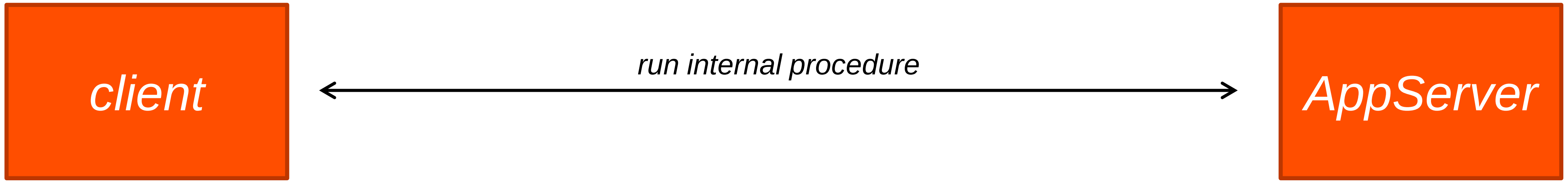
DELETE PROCEDURE hProc.

Support for Single-Run / Singleton - Persistent model

RUN server1.p ON hServer **PERSISTENT** SET hProc.



RUN internalproc IN hProc .

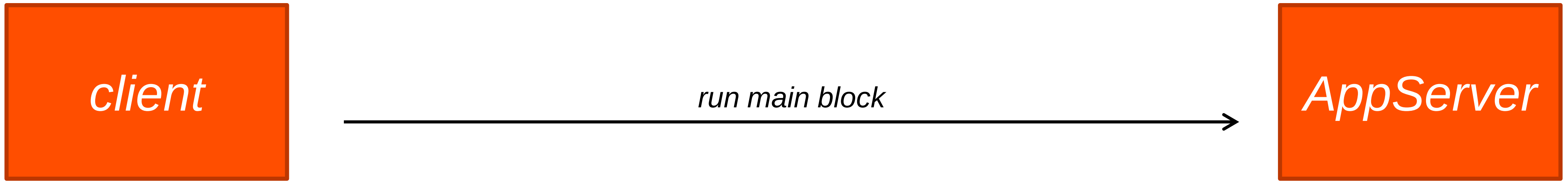


DELETE PROCEDURE hProc.

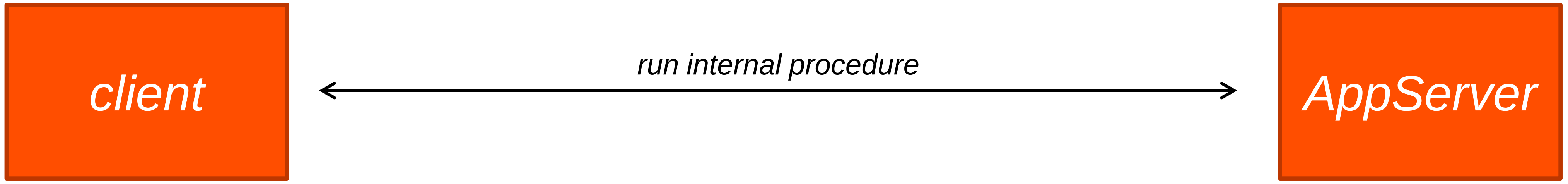
client
AppServer

Support for Single-Run / Singleton - Persistent model

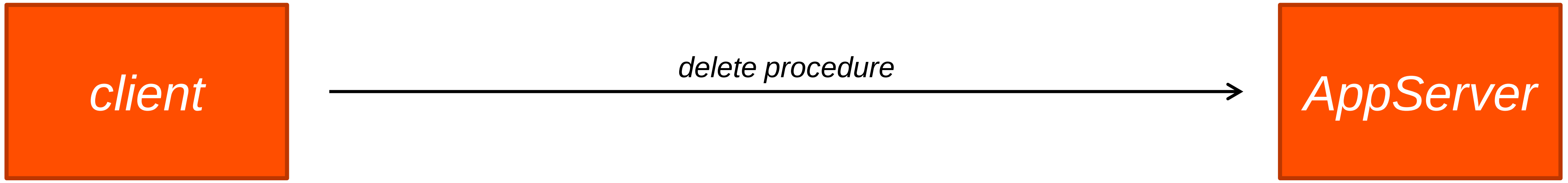
RUN server1.p ON hServer **PERSISTENT** SET hProc.



RUN internalproc IN hProc .



DELETE PROCEDURE hProc.



Support for Single-Run / Singleton - Single-Run model

RUN server1.p ON hServer **SINGLE-RUN** SET hProc.

RUN internalproc IN hProc .

Support for Single-Run / Singleton - Single-Run model

RUN server1.p ON hServer **SINGLE-RUN** SET hProc.

client
AppServer

RUN internalproc IN hProc .

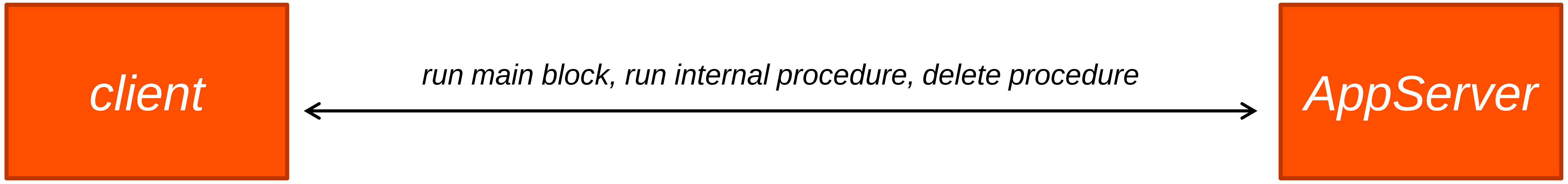
client
AppServer

Support for Single-Run / Singleton - Single-Run model

RUN server1.p ON hServer **SINGLE-RUN** SET hProc.

client
AppServer

RUN internalproc IN hProc .



Support for Single-Run / Singleton - Singleton model

RUN server1.p ON hServer **SINGLETON** SET hProc.

RUN internalproc1 IN hProc .

Support for Single-Run / Singleton - Singleton model

RUN server1.p ON hServer **SINGLETON** SET hProc.

client
AppServer

RUN internalproc1 IN hProc .

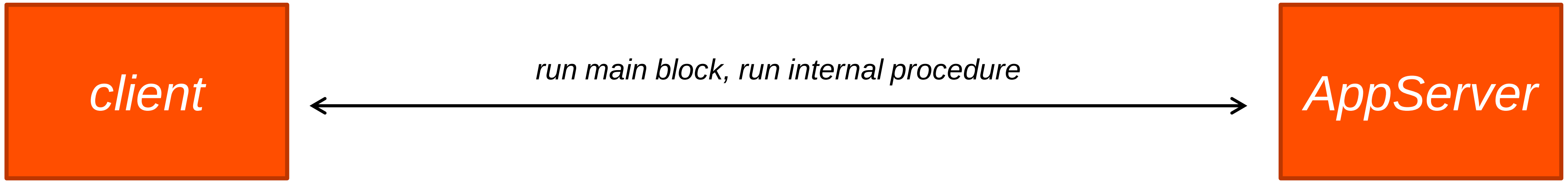
client
AppServer

Support for Single-Run / Singleton - Singleton model

RUN server1.p ON hServer **SINGLETON** SET hProc.

client
AppServer

RUN internalproc1 IN hProc .

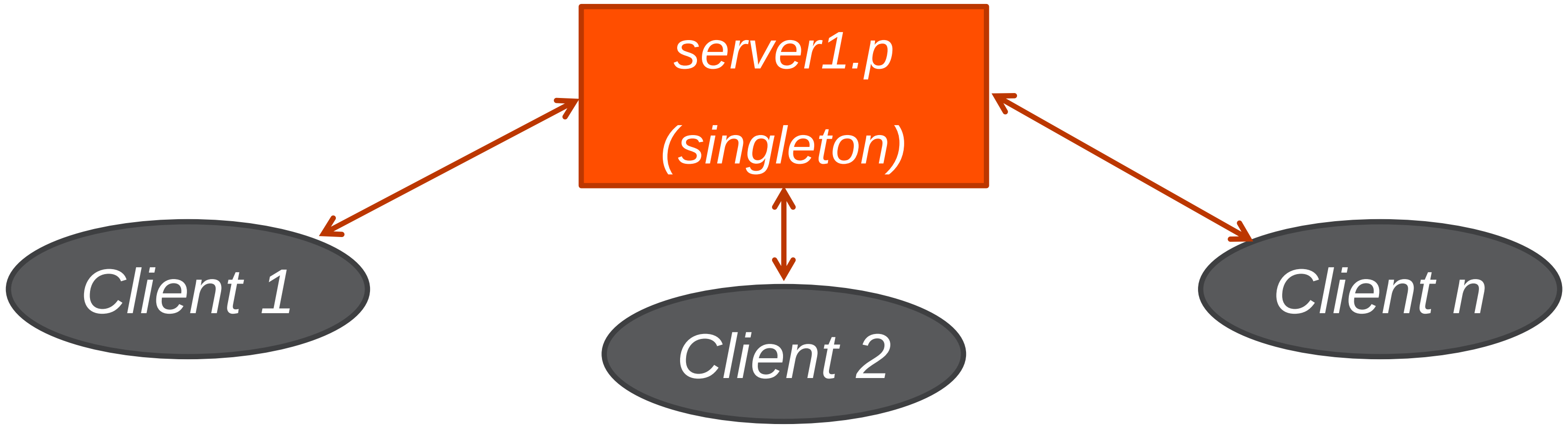


Support for Single-Run / Singleton - Singleton model

The singleton procedure is not deleted until the AppServer agent is shutdown.

Hint

Note: Once a Singleton procedure is instantiated on an AppServer agent, that procedure is used for all subsequent requests to that agent, even if requested by another client.

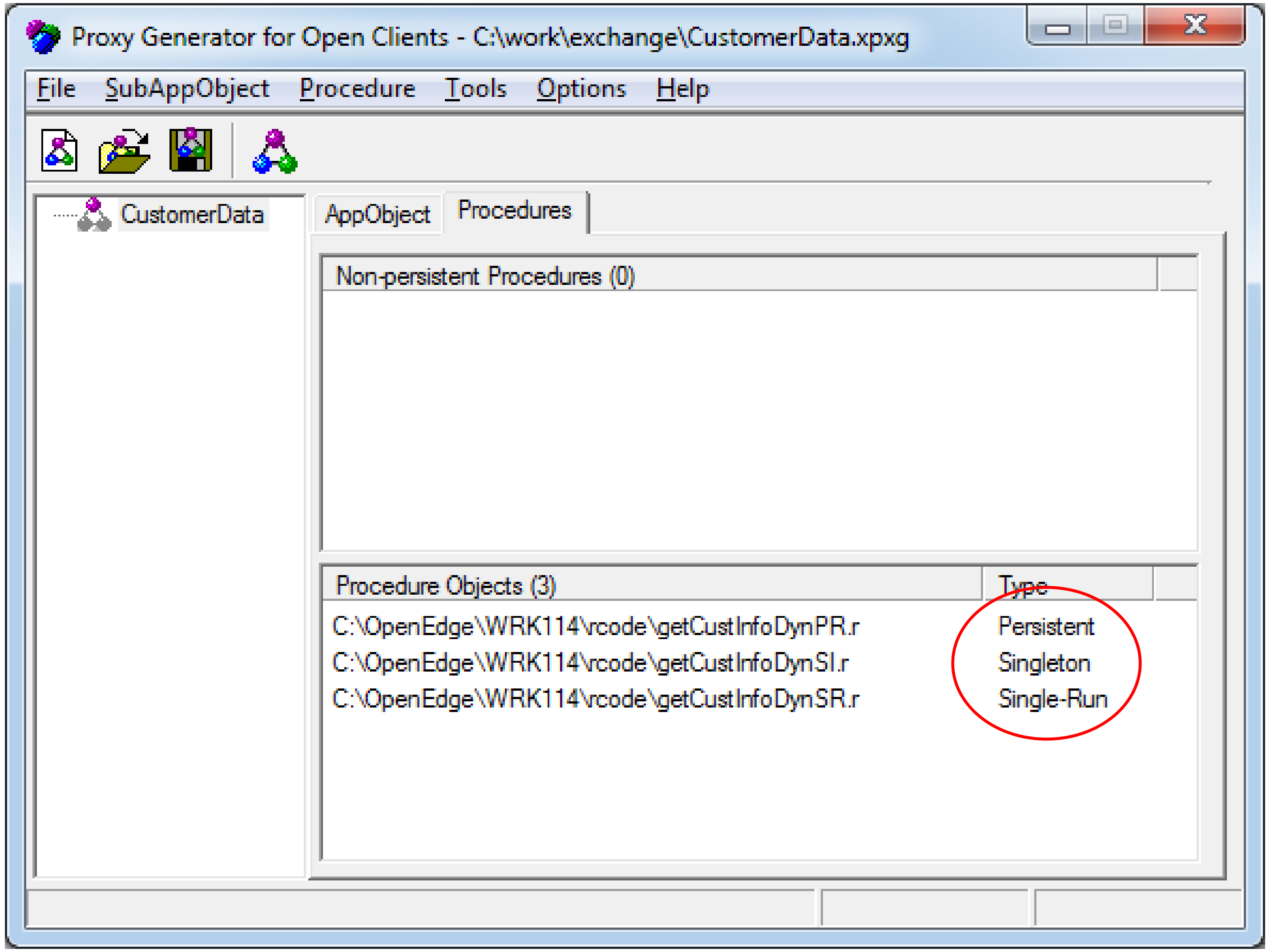


Support for Single-Run / Singleton

- New in 11.2
  - ABL client

- New in 11.3
  - ABL client Call Object handle
  - ABL client (running on Session, locally)
  - .NET & Java to OpenClient using ProxyGen
  - .NET & Java to OpenAPI

Support for Single-Run / Singleton


Proxy Generator for Open Clients - C:\work\exchange\CustomerData.xpxg
File SubAppObject Procedure Tools Options Help
CustomerData
AppObject
Procedures

Non-persistent Procedures (0)

Procedure Objects (3)	Type
C:\OpenEdge\WRK114\rcode\getCustInfoDynPR.r	Persistent
C:\OpenEdge\WRK114\rcode\getCustInfoDynSI.r	Singleton
C:\OpenEdge\WRK114\rcode\getCustInfoDynSR.r	Single-Run

Support for Single-Run / Singleton

- Java OpenAPI example

```
OpenProcObject genPO = genAO.createPO(“server1.p”,
ProcedureType.SINGLETON);

ParamArray params1 = new ParamArray(1)

params1.addInt64(0, lg1, ParamArrayMode.INPUT);

genPO.runProc(“internalproc”, params1);
```

Unicode Filename Support (Release 11.3)

Problem:

- I need to access files that have Unicode characters in their names from ABL.

Solution:

- Most ABL constructs (statements, widget methods, and functions) now allow filenames with Unicode characters.

- Examples: INPUT FROM, OUTPUT TO, editor:SAVE-FILE(), COPY-LOB, image files, OS-COMMAND, SYSTEM-DIALOG GET-FILE, xml-document:LOAD(), …
- Exceptions: Procedure files, class files, database files, log files
- Unicode printer names are also supported
- Recommendation: Use –cpinternal UTF-8 to ensure that the full range of possible filenames can be converted

Hint

What’s New in the ABL in Progress OpenEdge Release 11

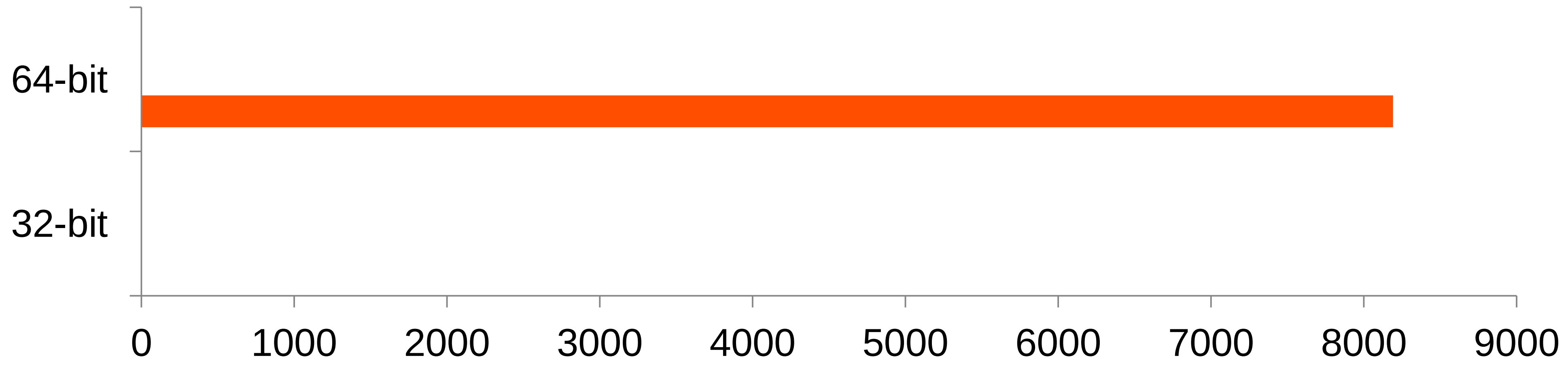
- Upgrades
- Security
- New Language Features
- **Windows 64-bit GUI Client**
- Questions & Answers

Windows 64-bit GUI Client (Release 11.3)

- What it is
- What it isn't
- Differences between 32-bit and 64-bit products
- Migrating an application

Windows 64-bit GUI Client

- What it is
  - Replaces the 32-bit GUI client which was included with 64-bit OpenEdge products on Windows
  - The 64-bit client allows access to a much larger address space (virtual memory) than the 32-bit client does: 2GB on 32-bit versus 8TB on 64-bit



Windows 64-bit GUI Client

- What it isn’t
  - It isn’t a requirement on 64-bit Windows systems
    - 64-bit versions of Windows can run 32-bit applications
  - It isn’t necessarily faster than 32-bit
  - It isn’t always a “plug-and-play” migration from 32-bit to 64-bit

Windows 64-bit GUI Client

- Differences between 32-bit and 64-bit products
  - The 64-bit client is called prowin.exe; the 32-bit client remains prowin32.exe
  - Report Engine is only 32-bit
  - WebSpeed development tools in AppBuilder can only access a local web server
  - Advanced Editing features are not available in 64-bit products

- Progress Developer Studio is available in both 32-bit and 64-bit versions

- You cannot install both the 32-bit and 64-bit versions of the same release on a machine
- You can install 32-bit and 64-bit versions of different releases on the same machine

32-bit 11.3 and 64-bit 11.3 – NO

32-bit 11.2 and 64-bit 11.3 - YES

Windows 64-bit GUI Client

- R-code is portable between 32-bit and 64-bit

- Am I 32-bit or 64-bit?
  - At run-time: PROCESS-ARCHITECTURE built-in ABL function
    - `IF PROCESS-ARCHITECTURE = 64 THEN …`
  - At compile-time: PROCESS-ARCHITECTURE preprocessor variable
    - `&IF {&PROCESS-ARCHITECTURE} = 64 &THEN …`

- OPSYS and WINDOW-SYSTEM
  - These values haven't changed:
    - OPSYS returns "WIN32"
    - SESSION:WINDOW-SYSTEM returns "MSWIN-XP"

Windows 64-bit GUI Client

- The 64-bit client supports these image formats:
  - .BMP – Windows Bitmap
  - .GIF – Graphics Interchange Format
  - .ICO – Windows Icon
  - .JPG – Jpeg
  - .PNG – Portable Network Graphics
  - .TIF – Tagged Image File Format

Windows 64-bit GUI Client

- External Procedure calls (DLL calls)
  - Compatibility:
    - The 64-bit client can only load 64-bit DLLs
    - The 32-bit client can only load 32-bit DLLs
  - Third-party DLLs – Ask the vendor for a 64-bit DLL
  - Homegrown DLLs – Port to 64 bit
  - Win32 API function calls should be reviewed in case parameter sizes have changed

Windows 64-bit GUI Client

- OCX Controls
  - OCX controls are mostly a thing of the past
  - Applications that use OCX controls extensively will need major work to run with the 64-bit GUI client

	32-bit	64-bit
Cihttp.ocx	✓	
Comctl32.ocx	✓	
Cscomb32.ocx	✓	
Cslist32.ocx	✓	
Csspin32.ocx	✓	
Mscomctl.ocx	✓	
Pstimer.ocx	✓	✓
Sstree.ocx	✓	

Windows 64-bit GUI Client

- GUI for .NET
  - .NET assemblies can be 32-bit (x86), 64-bit (x64), or both (AnyCPU)

Assembly Type	32-bit Client	64-bit Client
x86	✓	
x64		✓
AnyCPU	✓	✓

  - Infragistics assemblies are built for AnyCPU and work with both the 32-bit and 64-bit clients
  - Most third-party assemblies are also built for AnyCPU
  - Code that calls the Interop and P/Invoke services needs review

PROGRESS

What’s New in the ABL in Progress OpenEdge Release 11

- Upgrades
- Security
- New Language Features
- Windows 64-bit GUI Client
- **Questions & Answers**

PROGRESS