The 4GL in OpenEdge 11.x

(for small values of x)

Gus Björklund. Wizard. Progress.

PROGRESS software

Blame

- The following people have contributed parts of this talk
  - Evan Bleicher
  - Robin Brown
  - Fernando Souza
  - Gus Bjorklund
  - David Lund
  - Mary Szekely

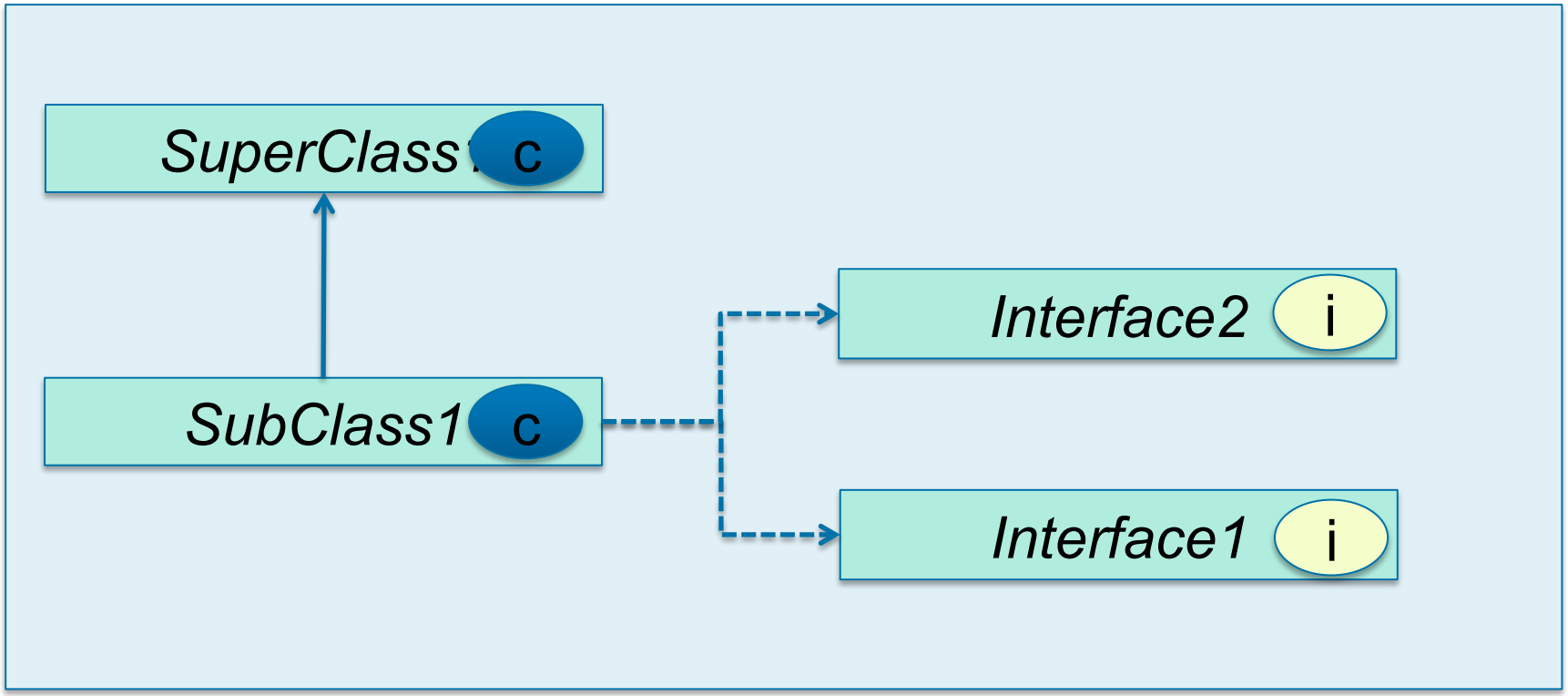
Categories

- OO4GL
- JSON API
- XML and JSON Improvements
- LIKE
- Authentication CallBacks
- Extend Capabilities
- VST for Temp Tables
- Performance enahncements
- 11.2 4GL stuff
- 11.3 4GL stuff

OO4GL

Interface Inheritance

- OO4GL Class type (*pre-11.0 functionality*)
  - Inherit from another class type
  - Implement multiple interface types



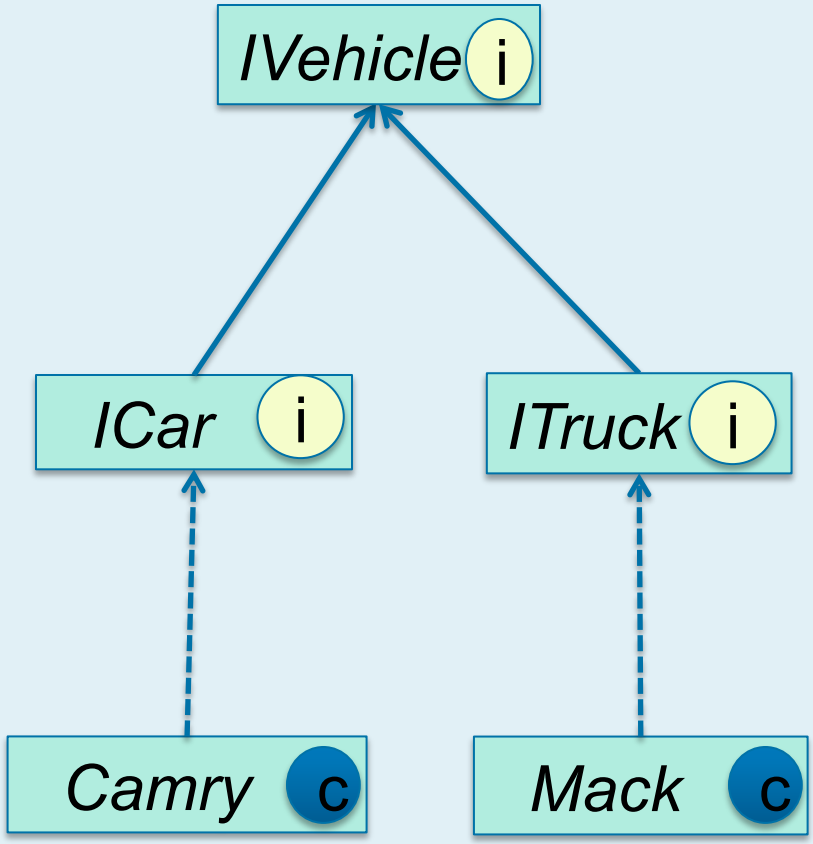
Interface Inheritance

- OO4GL Interface type - ***NEW!***
  - Inherit from multiple interface types
  - Inherit from .NET interface type
- Syntax

INTERFACE *InterfaceName*

[INHERITS *SuperName1* [, *SuperName2*] ...] :

Interface Inheritance



Interface Inheritance

- Program to the interfaces via polymorphism
- TYPE-OF function
  - Returns TRUE for Interface in hierarchy

```
PUBLIC METHOD LOGICAL handleVehicle(INPUT refVehicle
      AS CLASS IVehicle):
 IF TYPE-OF(refVehicle, "ICar") EQ TRUE
       THEN THIS-OBJECT:handleCar(refVehicle).
 ELSE IF TYPE-OF(refVehicle, "ITruck") EQ TRUE
       THEN THIS-OBJECT:handleTruck(refVehicle).
 ELSE THIS-OBJECT:handleGeneric(refVehicle).
```

Dynamic Properties

PROGRESS software

- Dynamically (*pre-11.0 functionality*)
  - Instantiate a class
  - Invoke a method
- Dynamically get or set OO4GL property - ***NEW!***
- DYNAMIC-PROPERTY function
- Progress.Lang.Class
  - GetPropertyValue
  - SetPropertyValue

Dynamic Properties

```
DEFINE INPUT PARAMETER myDynClassName AS CHARACTER.
DEFINE INPUT PARAMETER myDynPropName AS CHARACTER.
DEFINE OUTPUT PARAMETER mychar AS CHARACTER.
DEFINE VARIABLE  myClass AS  Progress.Lang.Class.
DEFINE VARIABLE  myObj AS Progress.Lang.Object.

myClass =
      Progress.Lang.Class:GetClass(myDynClassName).

myObj = myClass:New().

mychar = DYNAMIC-PROPERTY( myObj, myDynPropName).

/* OR */

mychar = myClass:GetPropertyValue(myObj,
                                  myDynPropName).
```

JSON API

JSON API – *NEW!*

PROGRESS software

- Built-in OO4GL classes for creating and parsing JSON
- JSON: JavaScript Object Notation
  - Lightweight data exchange format (without all that gross XML trash !)
  - http://json.org
- Use Cases
  - JavaScript Libraries supporting AJAX also support JSON
  - Which means that...
    - OE can easily become the back end of a Rich Internet Application (RIA)
      - AppServer
      - WebSpeed

JSON Data Types

PROGRESS software

- Four simple data types
  - string – “jump rope”
  - number – 17, 54.35, 0.9582e-42
  - boolean – true, false
  - null – null
- Non-standard data types commonly used
  - date/time – “2011-09-21T11:00:00-04:00”
  - binary – Base64 encoded string
- Complex data types
  - Object
  - Array

JSON Object

- Progress.Json.ObjectModel.JsonObject
  - Collection of name/value pairs
  - No order
  - Access by property name
  - Object surrounded by curly braces { }

- Can INHERIT from

{ “name-1” : value-1, “name-2” : value-2, “name-3” : value-3}

JSON Object Example

```
myObject = NEW JsonObject().
myObject:Add("name", "Dorothy Gale").
myObject:Add("age", 38).
myObject:Add("region", "Kansas, USA").
myObject:Write(myLongchar, TRUE).
```

myLongchar:

```
{
  "name" : "Dorothy Gale",
  "age" : 38,
  "region" : "Kansas, USA"
}
```

```
vChar = myObject:GetCharacter("name").
vInt  = myObject:GetInteger("age").
```

JSON Array

- Progress.Json.ObjectModel.JsonArray
  - Ordered list of unnamed values
  - Strict order
  - Access by array index
  - Surrounded by square brackets [ ]
- Can INHERIT from

[value-1, value-2, value-3, value-4]

JSON Array Example

```
myArray = NEW JsonArray().
myArray:Add(1).
myArray:Add(FALSE).
myArray.Add("jump rope").
myArray:AddNull().
myArray:Write(myLongchar, TRUE).
```

myLongchar:

```
[ 1, false, "jump rope", null ]
```

```
myArray:Set(2, 6.0).
vDec = myArray:GetDecimal(2).   /* vDec = 6.0 */
vLog = myArray:GetLogical(4).   /* vLog = ? */
```

Complex JSON Values

PROGRESS software

- Combination of simple values, objects and arrays

```
{
 "salesRep" : { "name" : "Dorothy Gale",
                "age" : 38,
                "region" : "Kansas, USA"
              },
 "tractorSales" : { "2009Quarterly" : [ 13, 27, 18, 9 ],
                    "2008Quarterly" : [ 11, 17, 32, 5 ],
                    "2007Quarterly" : [ 9, 25, 16, 10 ]
                  }
}
```

JSON API ProDataSet / Temp-Table Support

- JsonObject:Read ( )
  - Dataset
  - Temp-Table
  - Temp-Table Buffer
- JsonArray:Read ( )
  - Temp-Table
- READ-JSON ( ) Enhancement
  - New source types
    - JsonObject
    - JsonArray

XML and JSON

4GL XML and JSON Improvements

PROGRESS software

- Recognize more XML/JSON formats as DATASETs
  - Benefit
    - Easier integration with 3rd party products
  - PARENT-ID-RELATION
    - Child record has field with RECID of parent
      - PARENT-ID-FIELD
  - Availability:
    - READ-XML/SCHEMA ( )
    - READ-JSON ( )
    - bproxsdto4gl utility
    - bprowsdldoc utility
  - WRITE-XML ( ) / WRITE-JSON ( ) do the right thing

XML Example – PARENT-ID-RELATION

```
<CustomerOrders>
 <Customer>
  <CustNum>1</CustNum>
  <Name>Lift Tours</Name>
  <Order>
   <OrdNum>100</OrdNum>
   <OrdTot>234.89</OrdTot>
  </Order>
 </Customer>
 <Customer>
  <CustNum>3</CustNum>
  <Name>Hoops</Name>
  <Order>
   <OrdNum>200</OrdNum>
   <OrdTot>899.99</OrdTot>
  </Order>
 </Customer>
</CustomerOrders>
```

```
DEFINE TEMP-TABLE Customer
 FIELD CustNum AS INTEGER
 FIELD Name AS CHARACTER.

DEFINE TEMP-TABLE Order
 FIELD OrdNum AS INTEGER
 FIELD OrdTot AS DECIMAL
 FIELD Customer_Id AS RECID
    XML-NODE-TYPE "Hidden".

DEFINE DATASET
              CustomerOrders
  FOR Customer, Order
  PARENT-ID-RELATION rel1
    FOR Customer, Order
    PARENT-ID-FIELD
                Customer_Id.
```

4GL XML and JSON Improvements

- XML-NODE-TYPE “Hidden” / SERIALIZE-HIDDEN on DATASET
  - Root node maps to temp-table

```
<person>
  <name>Ken</name>
  <children>
    <child>Adam</child>
    <child>Elana</child>
  </children>
<person>
```

```
DEFINE DATASET personDset XML-NODE-TYPE “HIDDEN”
   FOR person, children, child
...
```

4GL XML and JSON Improvements

- WRITE-JSON method
  - New omit-outer-object argument

omit-outer-object = FALSE

```
{"tt": [
  {"f1": 11, "f2": 12},
  {"f1": 21, "f2": 22},
  {"f1": 31, "f2": 32}
]}
```

omit-outer-object = TRUE

```
[
  {"f1": 11, "f2": 12},
  {"f1": 21, "f2": 22},
  {"f1": 31, "f2": 32}
]
```

4GL XML and JSON Improvements

PROGRESS software

- Buffer-object:SERIALIZE-ROW ( ) – ***NEW!***
  - Serialize ONLY current row
  - Target-format “XML” and “JSON”

```
SERIALIZE-ROW
( target-format, target-type,
  { file | stream | stream-handle | memptr | longchar }
  [, formatted [, encoding [, omit-initial-values
  [, omit-outer-object ] ] ] ] )
```

Upgrade Xerces XML Parser & ICU Libraries

PROGRESS software

- Upgrade Xerces processor to the latest version
  - Upgrade to Xerces 3.1.1
- Upgrade ICU (International Components for Unicode)
  - Upgrade to ICU 4.8

Xerces XML Parser in OpenEdge 11.1

PROGRESS software

- In general, 4GL application programmers will not be impacted by the upgrade to the XML parser
  - Although not common, some users may see some differences

Areas where differences can be seen:

- Xerces 3.1.1 more closely adheres to the XML specification
  - In 4GL some areas may no longer pass as valid XML
- Some known issues are fixed by this upgrade
  - Customers that have encountered theses issues will benefit from the resolution
- Some error messages have changed

Do you LIKE it?

LIKE Phrase for Method and Function Parameters

PROGRESS software

- LIKE phrase Supported when defining:
  - Variables
  - Procedure parameters
  - Method parameters (new in OpenEdge 11.1)
  - Function parameters (new in OpenEdge 11.1)

LIKE for user-defined function and method parameters

```
CLASS simple:
  DEFINE VARIABLE myvar AS DECIMAL DECIMALS 3 FORMAT ">9.9999".
  METHOD PUBLIC VOID test (INPUT inch LIKE Customer.Name,
                           INPUT inint LIKE Customer.Cust-Num,
                           INPUT indec LIKE myvar):
    DISPLAY inch inint indec.
  END.
END.
```

```
Name            Cust-Num  myvar
--------------------  --------  -------
Lift Line               345  2.3490
```

Example of Using LIKE Phrase in a Function

```
/* Function prototype */
FUNCTION tstFunc RETURNS LOGICAL
         (INPUT pName LIKE Customer.Name) FORWARD.

/* Function implementation */
FUNCTION tstFunc RETURNS LOGICAL
        (INPUT pName LIKE Customer.Name):
  DISPLAY pName.
END.
```

User authentication callbacks

(e.g. user login validation)

4GL User-defined Authentication Using Callbacks

PROGRESS software

- OpenEdge 11.1 extends existing configurable user authentication to include customized 4GL user authentication

- This feature is visible to OpenEdge developers, as well as to those that distribute and administer 4GL based applications

So what has changed?

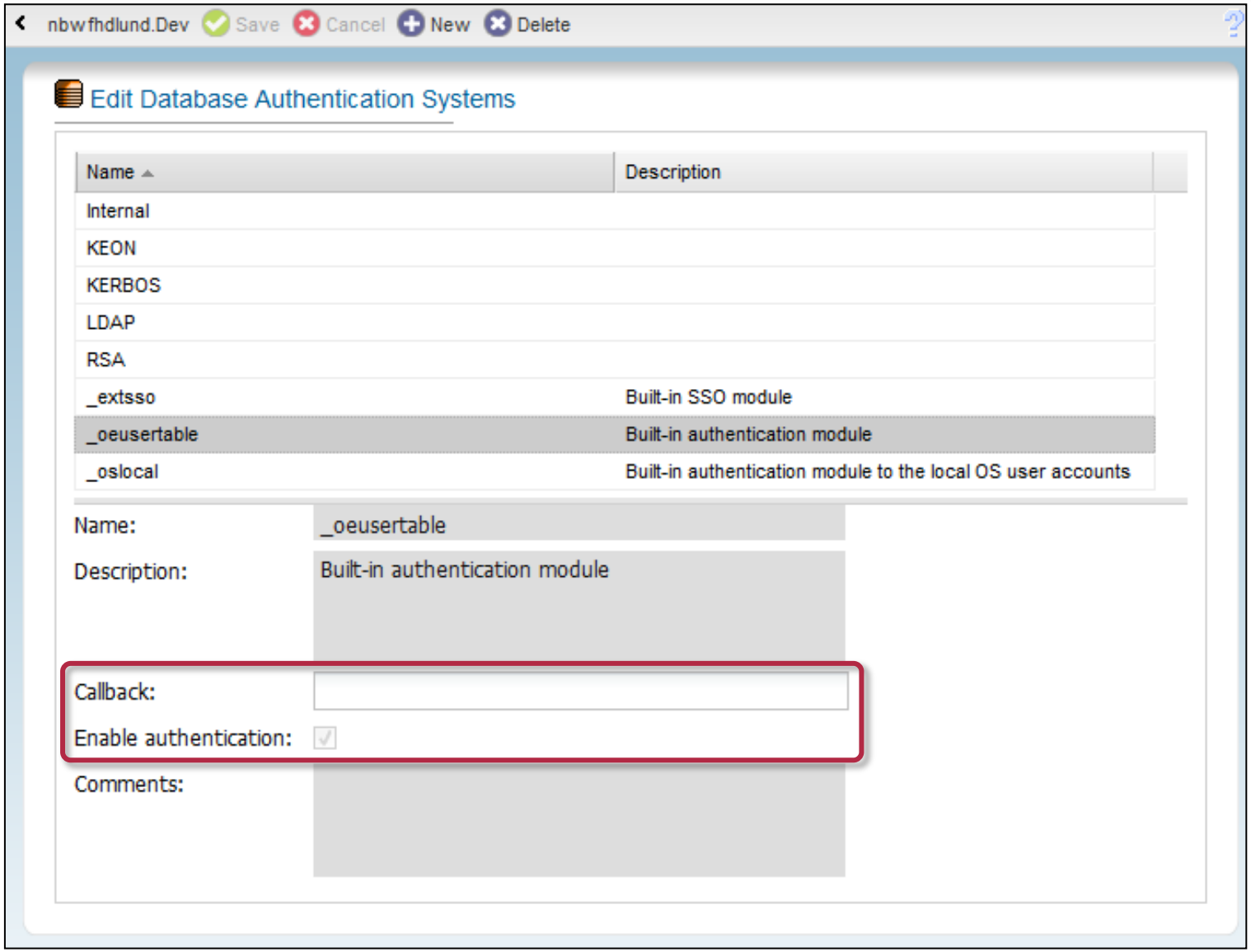
PROGRESS software

- The two inbuilt plug-ins provided earlier allowed for user authentication against the database's _user table and against local operating system accounts.
- Now we have an 4GL authentication plug-in mechanism that you can use to invoke your own code to do user authentication in whatever way you like.
- You use the new mechanism as follows:
  - create a client-principal object
    - and set values for various fields in it.
  - When you invoke either SET-DB-CLIENT() or SECURITY-POLICY:SET-CLIENT() then an entry-point in your previously registered 4GL callback procedure will be called.
  - Your code then examines the presented user identity, decides whether or not it is valid, and returns either an accept or reject return code.
  - Also CONNECT and SETUSERID()

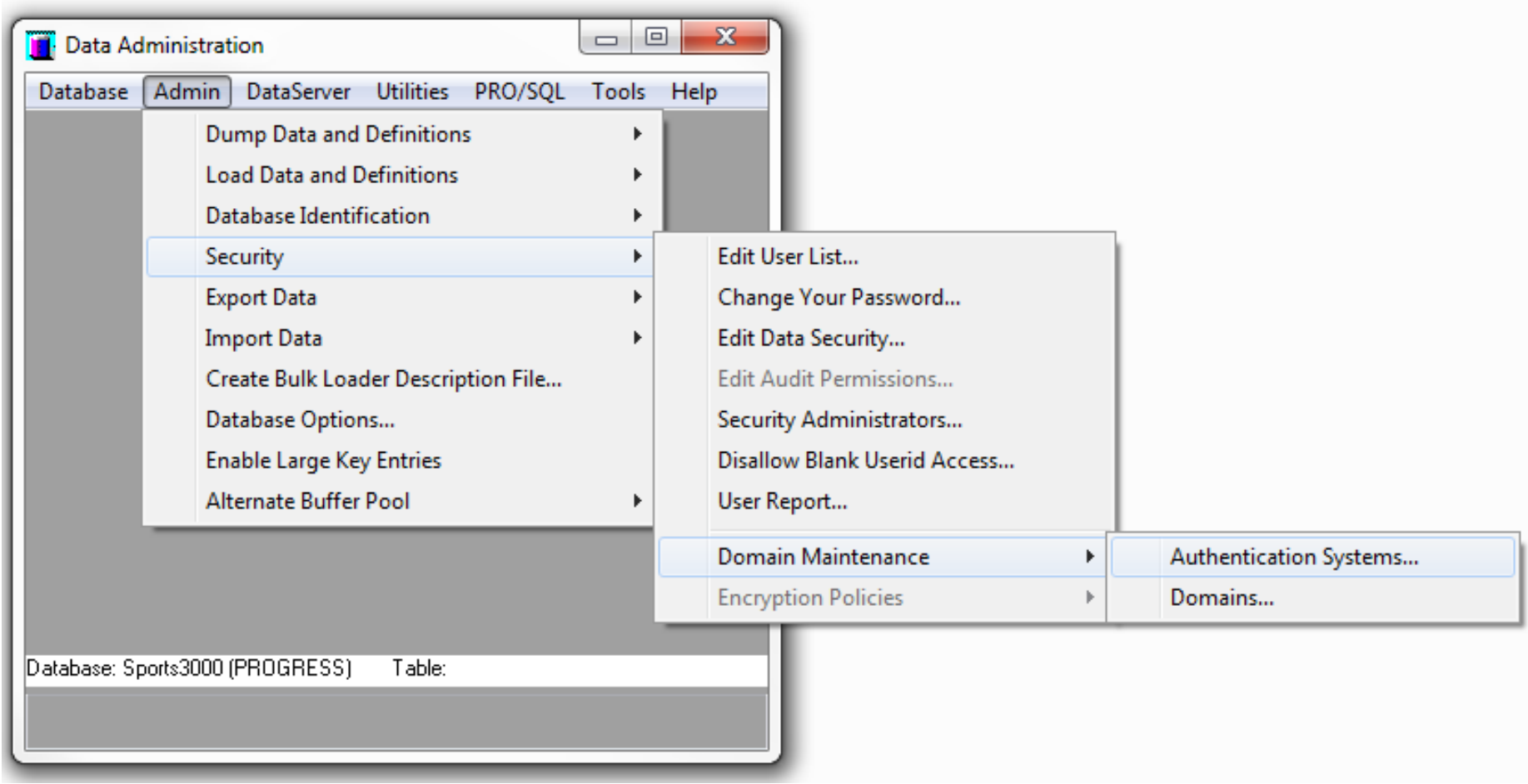
So what has changed? Part 2

- But wait . . . that's not all! You can also use the callback mechanism with the inbuilt authentication systems to extend those. For example, you can set additional values in the client-principal object or record all user logins somewhere suitable.
- To use this feature, all you have to do is set the callback procedure name in the _sec-authentication-system._PAM-callback-procedure* for those authentication domains in which you want a procedure to be called.
- You can find more information about this feature in Chapter 2 of the OpenEdge 11.1 manual entitled " OpenEdge Development: Programming Interfaces".

Edit Database Authentication Systems Page in the Database Administration Console



Accessing Authentication Systems using Data Administration



Overview of Callbacks

PROGRESS software

Three components of 4GL external procedure:

1. Main block
   - Run by OpenEdge
   - Executed (once) and completes before the callbacks execute
   - The main block cannot have parameters
2. AuthenticateUser procedure
   - Called during OpenEdge-performed user authentication
   - If a non-success error is returned authentication is denied and processing stops
3. AfterSetIdentity procedure
   - Called during user authentication and SSO
   - Errors are not returned
   - Does not have any impact on the success or failure of authentication

When the Callback is Called

Callback AuthenticateUser

- Called during OpenEdge performed user authentication process
  - For built-in systems: this is after the built-in system's authentication has completed and before the client-principal is sealed
  - For user-defined: at start of the user authentication process and before the client-principal is sealed

Callback AfterSetIdentity

- Called once each time OpenEdge completes setting a user identity (for SSO and OpenEdge-performed user authentication)
  - For each 4GL session and each OpenEdge database connection

AuthenticateUser and AfterSetIdentity Signatures

```
PROCEDURE AuthenticateUser:
DEFINE INPUT PARAMETER hCP AS HANDLE.
DEFINE INPUT PARAMETER cSystemOptions
                        AS CHARACTER EXTENT.
DEFINE OUTPUT PARAMETER iPAMStatus
                        AS INTEGER INITIAL ?.
DEFINE OUTPUT PARAMETER cErrorMsg AS CHARACTER.
END.
```

```
PROCEDURE AfterSetIdentity:
  DEFINE INPUT PARAMETER hCP AS HANDLE.
  DEFINE INPUT PARAMETER cSystemOptions
                        AS CHARACTER EXTENT.
END.
```

Working with the Client-Principal in Callbacks

- There are restrictions to operations that can be done in callback procedures
- For user-defined authentication the client-principal Primary-Passphrase is read-only
- Callback executing for a built-in system cannot read the Primary-Passphrase
- For a client-principal object you cannot:
  - Delete the object.
  - Call methods:
    - AUTHENTICATION-FAILED( ), EXPORT-PRINCIPAL( ), IMPORT-PRINCIPAL( ), INITIALIZE( ), LOGOUT( ), SEAL( )
  - Change attributes of:
    - DOMAIN-NAME, DOMAIN-TYPE, PRIMARY-PASSPHRASE, QUALIFIED-USER-ID, SESSION-ID, USER-ID

Progess.Security.PAMStatus Class

- Used by 4GL programmers to obtain security status codes that are returned by an 4GL authentication callback
- Callback procedure is called during the execution of an OpenEdge authentication or SSO operation
- Status codes are static class properties
- Sub-class of Progress.Lang.Object class
- Contains only a private constructor
- All properties have a data type of integer, and are static read-only publicly accessible values

Properties

- AuthenticationAccess
- AuthenticationFailed
- Custom
- InvalidConfiguration
- MaxTries
- MissingCredentials
- PermissionDenied
- Success
- UnknownUser

See: OpenEdge Development: OpenEdge Development 4GL Reference

Encoded 4GL Passwords

The purpose of encoded passwords is to:

- Not store clear-text password values in OS files (scripts, parameter files, and configuration files)
- Or pass them across a network connection

The encryption is not robust

Using Encoded Passwords

PROGRESS software

- AVM session's -P startup parameter
  - Command line connection to a database
- In a parameter file (.pf)
  - Set the -P startup parameter for a database in the .pf file
- 4GL Language
  - -P on the database CONNECT statement
  - SETUSERID function

  Authenticating the user identity of a client-principal
  - Client-Principal:PRIMARY-PASSPHRASE attribute
  - Client-Principal:Initialize( ) method
  - Client-Principal:Seal( ) method
  - Client-Principal:Validate-Seal( ) method

Creating an Encoded Password

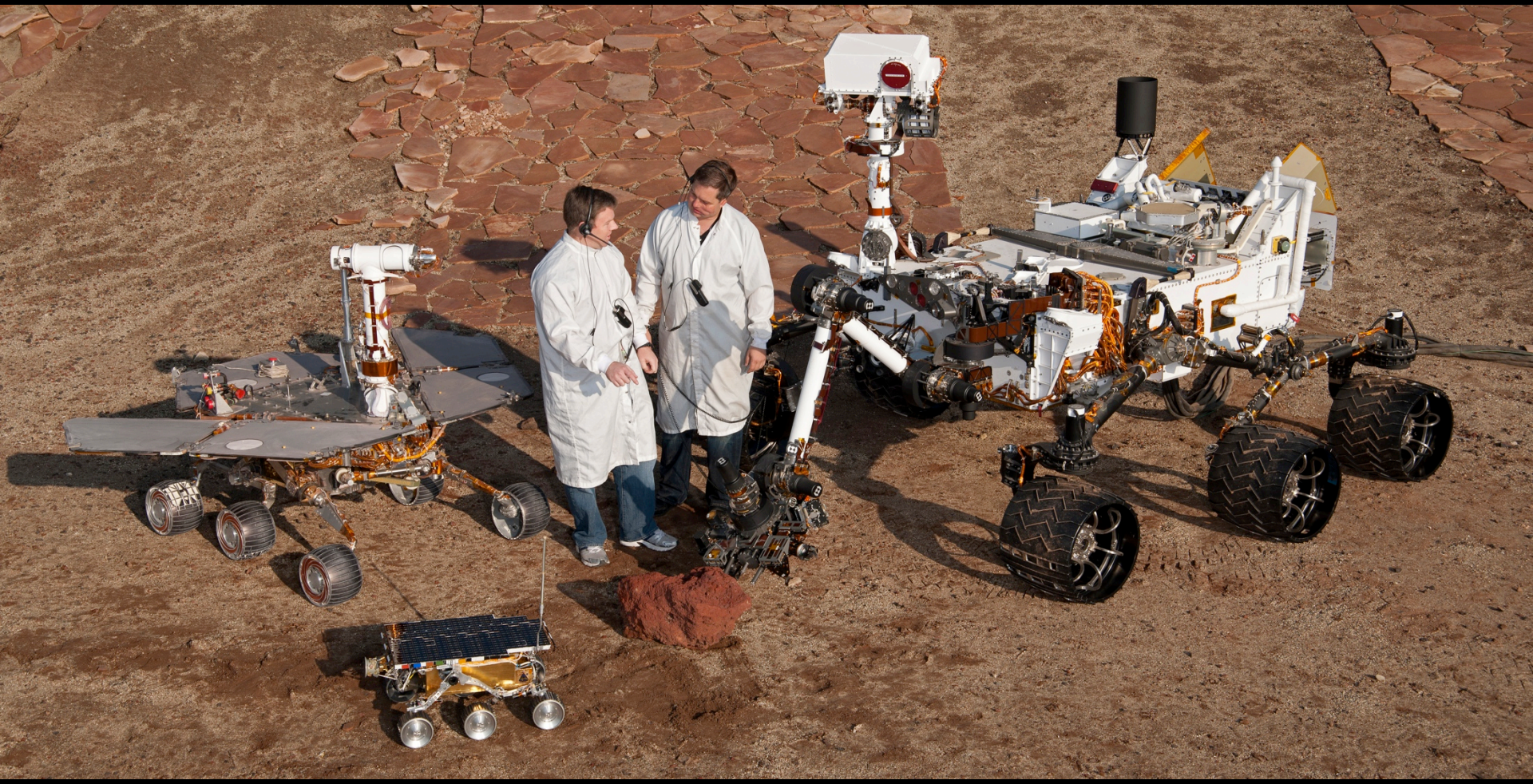
PROGRESS software

- Two ways to create an encoded password
  - Using AUDIT-POLICY:ENCRYPT-AUDIT-MAC-KEY method
  - From the command line – calling genpassword utility

Encoded passwords **DO NOT** prevent an intruder from using them to spoof a user account if they gain access to them. Encoded values must be part of a layered security scheme that includes the OS file system and/or 4GL application code.

```
encPwd = "oech1::" +
         AUDIT-POLICY:ENCRYPT-AUDIT-MAC-KEY(clearText)
```

```
proenv> genpassword -password "clear_text"
        xxxxxxxxx
proenv> echo "oech1::xxxxxxxxx"
…
```



Miscellaneous enhancements to various existing features

GUI for .NET

- .NET objects that are not related to UI are supported
  - Also supported in 10.2B02
- Non-user-interface executables support .NET objects
  - prowin32 –b (batch GUI client)
  - _proapsv (AppServer agent)
  - _progres (char client, WebSpeed agent)
- Updateable grids support LOBs
  - ProBindingSource Assign( )
  - 4GL
    - CURRENT-CHANGED, SAVE-ROW-CHANGES
    - BUFFER-COMPARE, Equality

Input Blocking Statements

PROGRESS software

- Input blocking statements (pre-11.0 functionality)
  - UPDATE, SET, PROMPT-FOR, CHOOSE, INSERT, WAIT-FOR, PROCESS EVENTS and READKEY
  - Restricted use, notably:
    - User-defined function
    - Non-void method
    - Or if user-defined function, non-void method are on call stack

Input Blocking Statements

PROGRESS software

- Input blocking statements (pre-11.0 functionality)
  - UPDATE, SET, PROMPT-FOR, CHOOSE, INSERT, WAIT-FOR, PROCESS EVENTS and READKEY
  - ~~Restricted use, notably:~~
    - ~~User-defined function~~
    - ~~Non-void method~~
    - ~~Or if user-defined function, non-void method are on call stack~~

- **Restriction removed!**

Suppress Warning Messages

- Suppress list of warning messages - ***NEW!***
  - -swl startup option
  - SESSION:SUPPRESS-WARNING-LIST
- Suppress all warning message
  - -sw startup option - ***NEW!***
  - SESSION:SUPPRESS-WARNINGS

Startup Options

- –inp
  - Maximum increased from 32,000 to 2,147,483,647
  - Also increased in 10.2B03
- -lkwtmo
  - Minimum decreased from 60 seconds to 10 seconds
  - small values may cause premature rollbacks

R-code Version Change for V11

- MUST recompile in version 11.0
- 32/64-bit compatibility restored (several times)
- Frame segment
  - 32 Kb to 4 Mb limit increase
- R-code header signature segment
  - 64Kb size limit lifted
  - Open Client (ProxyGen)
- Procedure library
  - Can now be > 4 Gb in size (but not with 32-bit releases)

Install

- WebClient
  - Uninstall a previous version
- License
  - Remove a license from a system
- Windows 64 bit packaging
  - add to included Win32 client
    - .NET
    - XML

VST's for temporary tables

Diagnostics - Temp-Tables

PROGRESS software

- Common questions:
  - How many temp-tables are in scope in the session?
  - Which program created each temp-table?
  - Most accessed temp-tables?
  - Growing DBI file. Why?

VSTs for Temp-Tables

PROGRESS software

- Temp-tables stored in DBI file (eventually)
- DBI = single-volume single-user DB
- Global data for DBI access
- Table/index statistics per temp-table is available (_TableStat, _IndexStat)

TableStat-id: 1	*read: 100*	*update: 2*
create: 1	*delete: 0*	*O/S Reads: 3*

-ttbasetable, -ttbaseindex, -tttablerangesize, and -ttindexrangesize

VSTs for Temp-Tables

PROGRESS software

- Built-in OO4GL Class

  Progress.Database.TempTableInfo

- Provide info on (static or dynamic):
  - Number of temp-tables in scope
  - List of temp-tables
  - Temp-table name
  - Name of procedure or class that instantiated it
  - Access to temp-table's handle

VSTs for Temp-Tables

- List of temp-tables in scope (including static temp-tables)

```
USING Progress.Database.*.
<…>
REPEAT i =1 TO  TempTableInfo:TempTableCount:
   TempTableInfo:GetTableInfoByPosition (i,
                                         OUTPUT hTable,
                                         OUTPUT cProcName).
   DISPLAY  hTable:NAME     LABEL "Table Name"
            cProcName       LABEL "Procedure Name"
            hTable:DYNAMIC LABEL "Dynamic".
END.
```

VSTs for Temp-Tables

- Built-in OO4GL Class

Provides access to the VSTs for temp-tables

```
USING Progress.Database.*.

DEFINE VARIABLE hVST AS HANDLE.

hVST = TempTableInfo:GetVSTHandle(VSTTableId:TableStatId).
```

VSTs for Temp-Tables

- Table/index statistics are lost when temp-table is deleted.
- Archive table and/or index statistics (from _TableStat and _IndexStat)
- Ability to log table / index statistics (new log entry type: TTStats)

Temp-table Logging

PROGRESS software

- New log entry type (Temp-Tables)
- Entries logged:
  - Creation
  - Deletion
  - Explicit EMPTY
  - Bind
  - Unbind

Created TEMP-TABLE ttCust (ID:1 Indexes:1) test.p @ 3

Deleted TEMP-TABLE ttCust (ID:1) test.p @ -1

- Basic statistics (number and size of record)

Performance optimisations

Delayed Instantiation

PROGRESS software

- Delay instantiation:
  - Temp-table
  - Associated Indexes
  - ProDataSet
- Improves procedure / class instantiation
- No coding
- Sample performance data (YMMV)
  - Class with 10 temp-tables – instantiation improved more than 50%

Table Scan

- Table Scan – used when accessing ALL records via FOR EACH
- Table must be in Type II storage area
- Does not use index to access
  - Access records sequentially
  - Data maybe in a different order than using an index

```
FOR EACH mytable TABLE-SCAN:
    totalCost = totalCost  + mytable.cost
END.
```

Other 4GL internal improvements

PROGRESS software

- No coding changes needed
- Improvements to class instantiation
- Improvements to method invocation
- Various improvements to 4GL runtime internals
- Fast table drop (type II area)

v 11.2 4GL stuff

Default scrolling startup option

- A 4GL query being resolved over a network
  - Is faster when the server can pack multiple result records into same network message (“prefetching”)
  - Happens when the query is guaranteed to ask the server for records in a forward motion (e.g., when `SCROLLING` is specified)
    - Note: Of course, the lock mode must be `NO-LOCK`
- If the query comes from a static `DEFINE QUERY` statement, where there is no `SCROLLING` keyword
  - It defaults to NOT keeping a client-side results list
  - It then uses the server’s index for prev/reposition type statements
  - Since forward motion cannot be guaranteed, the server prefetching is suppressed
- When `DEFINE QUERY` statements default to `SCROLLING`
  - A result list is always present
  - The possibility of network prefetching is guaranteed

PROVERSION

Pre-11.2	Displayed value
`PROVERSION`	11.1BETA

11.2 and higher	Displayed value
`PROVERSION`	11.2ALPHA
`PROVERSION()`	11.2ALPHA
`PROVERSION(0)`	11.2ALPHA
`PROVERSION(1)`	11.2.0.0.1171ALPHA

Enhanced structured error handling

PROGRESS software

- ON ERROR UNDO, THROW directive causes any error in the block to be thrown out of the block. If there is an associated CATCH block, it will execute.

```
DO TRANSACTION ON ERROR UNDO, THROW:

    . . .

END.

. . .

CATCH error-variable AS [ CLASS ] error-class:

     . . .

END [ CATCH ].
```

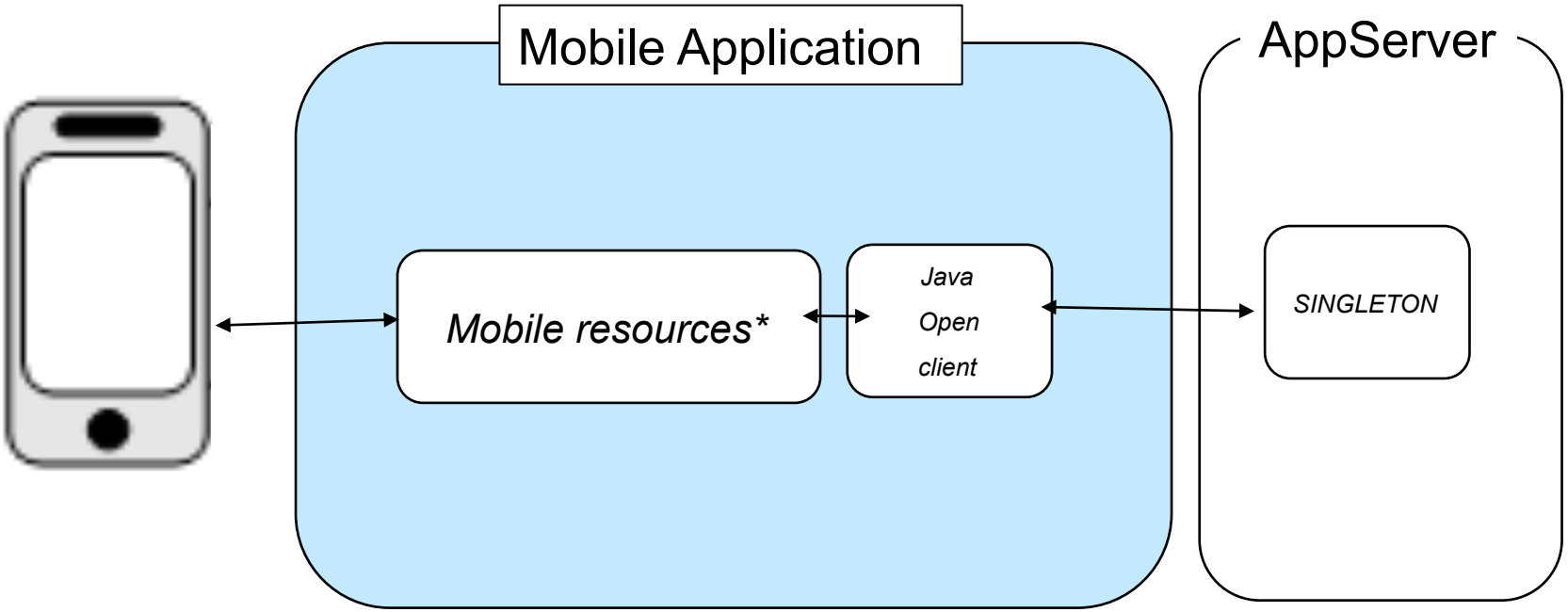
- The default error directive on “routines” (procedures, functions, methods and ON triggers) can be modified within a file using
  - `ROUTINE-LEVEL ON ERROR UNDO, THROW.`
- This can now also be done at block-level via the new statement:
  - `BLOCK-LEVEL ON ERROR UNDO, THROW.`
  - Changes the default for all blocks in a file that have a default error directive, including routine blocks, to have the `UNDO, THROW` error directive instead

New keywords: SINGLE-RUN, SINGLETON

- Two new keywords in the ABL are added to the RUN statement
  - `SINGLE-RUN`
  - `SINGLETON`
- Benefits
  - Reduce trips between client and AppServer for increased performance
  - Eliminates an AppServer agent from getting bound (or dedicated) to a particular client
- Users can take advantage of this feature with stateless and state-free AppServers but ***not*** state-reset or state-aware mode.
- Alternative to PERSISTENT

SINGLETON is key for Mobile*

- For Mobile, a [mobile] resource is the data and operations provided by one singleton class or procedure that is exposed on the client as a single OpenEdge JavaScript Data Object (JSDO).
  - This JSDO is roughly analogous to a ProDataSet plus any number of additional unrelated operations, as are made available in the singleton class or procedure.



Sub-second PAUSE

- Previously, the AVM rounded the fractional value specified in the "`PAUSE n`" phrase to the nearest integer
  - Example: For "`PAUSE 2.5`", the AVM would round the time-out interval value to 3 seconds (or 3000 milliseconds)
- `PAUSE` now allows the AVM to process a fractional value of `n`
  - Example: For "`PAUSE 2.5`", the AVM will now set the time-out interval value to exactly 2.5 seconds (or 2500 milliseconds)
  - Value is rounded to the nearest whole millisecond
    - For example 0.0015 will become 0.002, but Pause 0.0114 will become 0.001
- Other statements impacted are:
  - `WAIT-FOR ....PAUSE n`
  - `READKEY ... PAUSE n`
  - `CHOOSE.. PAUSE n`
- Customers requested this capability to improve how they write batch processing tasks in the ABL

Note: Do no expect a high degree of accuracy especially for values less than 0.1 second

Coming soon to a computer near you:

v 11.3 4GL stuff

(maybe)

11.3 stuff

- soap 1.2 support
- Unicode filenames for Windows
- dynamic access to ooabl inbuilt objects
- shorter *class1():bar* instead of *(new class1()):bar*
  [cuz Julian whinged so well]
- PROCESS-ARCHITECTURE function
- DISPLAY-TYPE
- Windows 64 bit GUI client

Not enough ?

Do you need more ?

Need more?

- Any OpenEdge customer may submit an enhancement request by sending email to:
  openedge-enhancements@progress.com
  and including the following:
- A description of the desired enhancement, specifying what should be changed, deleted, or added.
- Tell us the PROBLEM you want to have solved, NOT a supposed solution to some unstated problem.
- Tell us in what circumstances the problem arises and what effect it has on your code/application/system.
- Tell us what workarounds, if any, can be used as interim solutions. Also tell us why these workarounds are insufficient.
- Do not request existing features.
- Note: Please write your enhancement requests in *English*. OpenEdge development group cannot read other languages.

Questions

email: gus@progress.com