OpenROAD 6.2 – The New Features, from a Client Perspective

New in OpenROAD 6.2 – For all users of OpenROAD.
First of three presentations.
Sean Thrower
May, 2015
Disclaimer

This document is for informational purposes only and is subject to change at any time without notice. The information in this document is proprietary to Actian and no part of this document may be reproduced, copied, or transmitted in any form or for any purpose without the express prior written permission of Actian.

This document is not intended to be binding upon Actian to any particular course of business, pricing, product strategy, and/or development. Actian assumes no responsibility for errors or omissions in this document. Actian shall have no liability for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials. Actian does not warrant the accuracy or completeness of the information, text, graphics, links, or other items contained within this material. This document is provided without a warranty of any kind, either express or implied, including but not limited to the implied warranties of merchantability, fitness for a particular purpose, or non-infringement.
OpenROAD 6.2 – new features addressing customer requirements

- Features illustrated in the presentation will require the first OpenROAD 6.2 patch
  - p14746 or later
- This is the first of three presentations covering OpenROAD 6.2

p14746 OR 6.2.0 (int.w32/00)
In addition there is the usual long list of client-driven enhancements.
**DBMS interaction**
- Inline SQL, QueryObjects
- Powerful model of data-mapping, separating data from field

**4GL**
- Event model considerably superior to rivals

**OO**
- Endusers’ OO “work-model” will map 1:1 with OR’s implementation: single inheritance, single interface, 2-level overlapping classes. Limited datatypes though

**Field-Property model**
- No provision for rounding, asymmetry, sectors and sprites, intrinsic animations

**Resources**
- Binary files (docs, software, data, applications)
- Pre-defined scope-escaped functions, dynamic code, lists and images
- Comments
- Ancillary properties, including state and “informal” data
Meeting the OpenROAD 6.2 Objectives ...

- Providing explicit support for some key business requirements
Under the first category comes the detailed simulation of Windows 7 styling, not only for new frames, but even existing applications; the W7Styler will convert all the frames in an application without needing to change a single line of code.

Other first-category changes are direct support for visual allocation systems (airline or hall seating, hotel rooms, etc); for "active" maps and diagrams, including irregular regions; and for the automatic generation of rich business classes and functional frame displays that can be used as-is or readily developed further to supply or replace the many bread-and-butter enduser frames that production applications include.

The first five of these features in particular must be, and are designed to be:

- directly useful
- easy to integrate
- easy to customize and extend by clients and by Actian
- easy to cannibalize
- easy to understand

... as well as actually being needed by clients!
Numbers in circles in the title refer to the numbering of the sub-objectives on the previous page

**Demo manager:**
(Running this before the presentation displays a “topmost” toggle-menu for all of the demos listed in this presentation’s Notes sections)

w4gldev rundbapp remotehost::or62demos d201504_rundemos
Demo:

d4gldev rundbapp remotehost::or62demos propertychanger –cw7styler

Default all options except application (choose D201504_VIDEOS)

• The conversion takes around 2 minutes

Then compare the restyled application with the old (_BAK) version:

d4gldev rundbapp remotehost::or62demos D201504_VIDEOS_BAK
d4gldev rundbapp remotehost::or62demos D201504_VIDEOS

Use Ctrl-Tab to move the inputfocus onto the buttonfields to show pulsing. Mouse over the buttonfields and tablefield headers to show highlighting.
Restyling the old Videos application:
• no code change – not a single line.

Same result when tested on various Workbench applications, and client samples.
Demo:
D201504_VideosConverted
Run application
Choose Check Out option
Enter 151 as customer account
Ctrl-Shift-Tab to move focus to “Commit Changes”
Hover mouse over Date Out column header

Demo:
D201504_BitmappedTabfolderTabs -cBDPTabHighlighting
See also Videos conversion.

Restyling applied to ButtonFields, EntryFields, TableField headers, TabFolder tabs, SubForms, other compositefields, FreeTrims, Mainbars, RectangleShapes, ControlButtons.

Most other fieldtypes are already Windows7-like style, since we used native widgets for them.

Font changed to Segoe UI 9
Demo: D201504_DefinedResponses_Sprites – cprogressbars_coded

Marquee bars can be either “chase” or “zigzag” in appearance

Other points (covered in second presentation):
- The difference between chase and zigzag bars is just a handful of lines of 4GL code in the setup (no code is required at runtime).
- Progress bars need the 4GL to tell them what value to display, and when; no other runtime code is required.
- The graph itself is pure OpenROAD, no code other than the x,y calculations and a couple of system method calls
Restyling features – Other styles

- Windows7 is certainly the most complex of the current styles:
  - Each form Button has 10 different appearances, involving:
    - Pulsing and highlighting; double gradients; varying outlines (1, 2 and 3-layer concentric, 3D); varying roundings
  - Tabfolders, ListViewFields and other tabular displays have mouseover-highlighting headers
  - Other fields have special features (4-color independent outlines for text fields)

- But OpenROAD 6.2 can simulate virtually all of these effects
  - just using the new generic features (no Windows7-only features at all)

- Which means that PropertyChanger conversions to other Windows or non-Windows styles can also be created using these features
  - Probably more easily
  - Adapting and pruning the “Windows7” PropertyChanger code
Generating Userclasses and Displays

Essential features of Userclass generation:

- Must generate "real" overlapping navigable business userclasses
  - Simple table-mapped userclasses are for DBAs, not endusers!
  - All generated classes must "make business sense" to endusers
- Must generate full database interaction for each class
  - Allowing for different DBMS (as far as possible)
- Must generate richly-defined class interrelationships
  - Generated Cows must have Head and Legs attached!
- Must capture each attribute's category, subtype, order, and values
  - Otherwise all these things have to be ad-hoc coded, frame by frame
- Must be customizable by clients, during the generation process
  - Because "the same" business object is different in different organizations
- Must be customizable by clients, when editing the finished product
  - Too often, automated generation makes the product almost impossible to work with afterwards

?! That's not a cow...

... but that is!
Demo:

```
w4gldev runimage workbench.img -Tall -/appflags profile=or62demos application=d201504_frequentflyer
Create and edit User_profile userclass
Choose Attributes | Insert | From Database Table
Continue as shown
```
Generating Userclasses...
The TaggedValues arrays for the userclass and for each property contain much useful ancillary information,
And the userclass’s Queries array contains ready-to-run QueryObjects
Generating Displays

Essential features of Frame Display generation:

- Must generate "real" overlapping navigable frames
  - With enough detail for enduser to make work decisions without opening another frame
  - As many tablefields as are needed to provide "enough detail"
- Must optimize the layout for "sense" (address fields together in right order) and for size. For example:
  - Same-category attributes displayed in same panel; Identity panel always topleft
  - Tablefields on different tabfolder pages (optionally)
- The generation of the display must be customizable by client developers
  - Because many attributes are not actually displayed in a business-specialized frame
- The finished display after generation must be customizable by client developers
  - Too often, automated generation makes the product almost impossible to modify afterwards
- Should also offer a "working version", not just a passive display
  - With select, populate, change, navigate support based on a frame template.
Demo:

```bash
w4gldev runimage workbench.img –Tall /appflags profile=or62demos application=d201504_frequentflyer
```

Create and edit `User_profileDetails` frame from the `active_display` frametemplate

Choose Insert | Display from User Class

  The Select a User Class dialog appears

Select `d201504_frequentflyer` from the Application list

Select `user_profile` from the User Class list

  Note that there are rich Advanced and Customize options to tune the choice of fields to display, their properties, and the overall layout, and which capabilities to include (interframe navigation, for example)

Click OK and wait for the display to be generated

  The generated frame shows the userclass attributes as fields, grouped into panels by their category (identity fields, location fields, contact fields, etc), with object panels on the right hand side (Airport). Tablefields are at the bottom, as pages of a tabfolder (most business objects in the real world have several array attributes, not just one or none).

  Field order is preserved, so that name fields and address fields are in the right order.

  The size of the frame is as close as possible to whatever size you gave the Frame Editor.

<continued on next slide>
Demo (continued):
Hit F5 to run the frame
Rightclick the Open (second) toolbar icon and choose the Selected Item option
Doubleclick the sample@ingres.com entry
  The corresponding user is displayed
  Note that the frame can be edited and the changes saved
Leftclick the Open (second) toolbar icon
  All the users are selected, and the first one displayed
Click the Next (ninth, right-arrow) toolbar icon repeatedly
  Each user in the set is displayed in turn
Click the Menu icon in the topright of the Airport panel and choose the Open option
  The generated Airport frame is displayed, populated with the airport details, including the lists of arrivals and departures, both the scheduled ones and the actual flights this user has taken
  Note that the generation process is completely generic - you would need to change the Airport frame tabs to Flight Arrivals, Flight Departures Scheduled Arrivals, Scheduled Departures.

Each generated frame contains about 60 lines of code – the RequestManager helper class in Core does all the real 4GL work, including database interaction, navigation, item selection and traverse. Each of these functions can be overridden in the frame by a local procedure.
RequestManager is a UserClass, so it can be inherited and customized.
Using the Mapping Functionality

Current interfaces are increasingly keyboard-free

- More focused, more accurate, and faster for endusers
- Often requiring selection of irregularly-shaped or -placed visual elements
  - Mapping and tracking systems (maps, diagrams); allocation systems (seating, rooms)
- Simple to implement in OpenROAD, once we have irregular-shape selection (and associated drag-drop)

Essential features:

- Must have mouse-select (by click or hover) of irregularly-shaped or -placed items
- Must have mouse-drag of such items with notification of the 4GL.
- Must be scaleable and efficient
  - Allocation systems and maps involve hundreds of visual items
- Must be easy for developers to setup and to code
Demo:
This demo uses the new setup-frame capability, to minimize runtime code

```
w4gldev runimage workbench.img -Tall -/appflags profile=or62demos application=d201504_imagemapping component=countymap, command=open
```

Select the SetupName entry in the Property Inspector
The Setup Frame dialog will appear
Select the "D201504_Imagemapping" application and the "setupcountymap" frame, and click OK
The SetupCountyMap setup frame will run
Click the "County Outlines" tab
An outline map of English counties will appear
Click the "Setup the county map" button
After a few seconds each county will be coloured a different shade of grey
Close the setup frame
Note that the setupframe has analysed the data it has and has setup the CountyMap frame to run whenever the enduser needs it.

<instructions continue 2 slides on>
Running the Mapping setup-frame

- Prepares the County Demographics frame
  - The setup frame is use-and-lose
- All done using tagged values and new bitmap and stringobject methods
  - Code is entirely generic 4GL
  - Data is simple and freely available online
Demo (continued):

Run the CountyMap frame
Click any point in SouthEast England on the satellite image
  The county under the mouse is outlined in green, the name and demographic data for that county appear on the right, and a satellite image of that county appears above the data
Click outer points in SouthEast England to show different counties and their data
  Note that this is a genuinely useful facility, presenting genuinely useful data, using a simple generic mechanism adaptable to a wide range of business cases.

Select CountyMap, then choose File | Revert to last saved
Dynamic Tracking Systems

- If we add OpenROAD 6.2 Sprites to the Mapping capability...
- We can implement dynamic tracking systems
  - Flight tracking
  - Transport/delivery tracking
- With all the same advantages of simplicity, scaling, and ease of use
Demo:
D201504_AllocationSystems -cAirlineSeating
Drag the “yellow” passenger to one of the available seats, and drop the passenger there
See next slide for result:
- The passenger now occupies that seat, the previous seat is now available, a status bar message identifies the new seat.

One field, 1 bitmap, 4 sprite images. There are no “passengers” on the initial bitmap.
Business data in databases gets out of step as input and application errors cumulate

- Fixing this is a job for the DBA (or Data Administrator), not the end user
- Developers need to provide frames dedicated to data repair/administration.

Essential features:

- Must be tabular, so that the context of a change is more apparent
- Must offer CRUD capability
- Must be single table
  - Multi-table displays frequently confuse the administrator, and prevent certain kinds of repair.
- Must be reasonably* scaleable
  - *Some business tables have hundreds of columns and millions of rows, but manual correction would be entirely inappropriate for these
Demo:

w4gldev runimage workbench.img –Tall -/appflags profile=or62demos application=d201504_outofbox

Create frame AirportMaintenance using table_editor template
- A table-selection dialog appears
  Specify “airport” table from the table list and click OK
  A tablefield-based display is created, whose columns correspond to the airport table’s columns

Save the frame
<instructions continue on next slide>
Demo (continued):

Choose File | Open

The table populates with all the data from the database airport table

Select one or many records

Selected records have red text, and the current row is marked with a red asterisk

Scroll the data

Insert a row (using the File menu or the toolbar icons)

Type new data into the added row

Save the new record

Close and reopen the frame and choose File | Open and scroll if necessary to confirm that the record you created is in the database

The added record is there

Delete the record and save

Close and reopen the frame and choose File | Open and scroll if necessary to confirm that the record you deleted is no longer in the database

The added record is gone
✓ Supplying key business requirements

1. Restyling to up-to-date look-and-feel(s)
2. Same-code (unchanged code) transformations
3. Generated classes and displays
4. Active-map and allocation (room, seat, ...) capabilities
5. Richer out-of-the-box capabilities
6. Easier partitioning – covered later in this presentation
7. Easier deployment – covered later in this presentation
OpenROAD 6.2 introduces many generic enabling-changes, including:

- New bitmap, string, field, helper-class methods
- New intrinsic behaviors
- More system classes saved with components
- Prestored behaviors + setupframe
- Custom restyling support
- Useful mapping support

These changes directly address the cost-reduction requirement. In particular the ability to prestore TaggedValues and Defined Behaviors, and the frame-handling features that RequestManager provides, enable dramatic simplification and reduction in the 4GL code required to manage frame displays, and its destabilizing effect on other frame code. It is now possible and straightforward to provide rich custom restyling of existing applications without changing the existing code: Windows 7–like restyling is provided as a click-driven conversion out of the box; other restylings are equally possible to provide. The mapping support enables straightforward implementation of seat and room-allocation systems, and useful implementations of simple geographic-location systems.
The result to be
• More exactly correct
• More lowcost-extensible
• More lowcost-maintainable
• More saleable to management (internal and external)
• More timely

The development to be
• Less risky
• Easier to implement
• Less skill required
• Less rollback cost
• Smaller
• Less copy-paste-amend
• Less compromise required
• Less “provideds” (provided all frames look like X, provided architecture is master-detail, etc etc)
• More testable
• Less bug-prone
Must be directly useful, easy to integrate, easy to understand. Must make a difference to the cost not only of new development and use of the new features, but also ongoing maintenance of existing features.

Should reduce the number of coded events needed to achieve a single effect.
Enable removal of runtime setup code using Setup Frames

- Existing applications have a lot of “setup” code in each frame
  - Some of the code does need to be there; but other code was there because before 6.2 there was no practical way to prestore the setup – now, there is.
- The Property Inspector now has as SetupName option
  - Specify your setup frame, and it will be called from the frame editor
  - Code your setup frame to execute the setup changes the target frame needs
    - You can specify a series of setup frames, each doing a different setup task
    - Each target frame’s setup history is recorded
  - When the target frame is saved, the setup changes are in place, so all that setup code is no longer needed in the runtime
    - Don’t confuse this with FrameTemplates, which have an altogether different role
- Changing your existing applications to use this is straightforward:
  - Move the frame’s setup code into local procedures, test;
  - Move the procedures into a setup frame and apply it, test …

Numbers in circles in the title refer to the numbering of the sub-objectives on the previous page.

The new 6.2 capabilities will increase the setup-code requirements …
Demo:

w4gldev runimage workbench.img -Tall -/appflags profile=or62demos application=d201504_imagemapping component=countymap, command=open
Continue as shown
The new Setup Frame facility saves custom runtime code and reduces startup time. Setup Frame lets developers predefine frame setup code and put all setup code in single, coherent locations separate from the runtime application. All the changes to the frame are applied in advance so that when a frame starts up at runtime, everything is ready to go – significantly reducing startup time and reducing the complexity in managing frame setup.

Not only is the setup code no longer contaminating the frame at runtime, but it doesn’t need to be tested any more as part of bug-fixing, upgrades, etc. Even the initial testing is simpler, since it doesn’t have to be “foolproof”, it just has to work correctly when used correctly.
What data and objects can be stored in OpenROAD? Prior to 6.2, all saved, imaged or exported applications could contain:

- fields, lists, trees, strings, components

The applications can now also contain:

- tagged values, hashtables, longbyte objects, prochandles, arrays

... which means that imaged, saved, and exported applications can now contain data of any size, kind or complexity
What are these Defined ("ready-to-use") Behaviours?

- InputEvents-Response elements
  - InputEvents are not 4GL events, but mouse actions, key actions, user actions, and timed alerts
  - The Responses are usually visual changes, but can be 4GL procedure calls

- **25** inputevents
- **40** properties, full range of sprite behaviours, user-defined 4GL procedures, multiple simultaneous actions, etc

- Fields respond to InputEvents only if they are enabled for them
- No 4GL event blocks are involved
… Enable storage of ready-to-use behaviours and items ...

- Pre-storing the Windows-7-like styling in OpenROAD 6.2 saved another 500 lines of runtime code (in addition to the Setup savings)
  - and made the processing simpler and more reliable

No 4GL code or events required
Examples of Hansel and Gretel code / Adhoc trails:
- passing an incrementing counter to a userevent as the MessageInteger
- including the datatype of a variable in the variable’s name
TaggedValue arrays are attached to, and saved with, all fields, attributes, components, applications.

There are TaggedValue-manipulation methods in each of these classes.

Each TaggedValue can hold (and save) keyed data and objects, in any combination or amount.

Almost all useful System Class objects can be saved, but not User Class objects.
Demo:
D201504_DefinedResponses_Panel -cPassportDetails
Hover mouse over any “?” sprite
the infopanel frame appears alongside the sprite, displaying that input field’s description.
Move the mouse off the sprite
the infopanel frame disappears.

Demo:
w4gldev runimage workbench.img –Tall -/appflags profile=or62demos
application=d201504_definedresponses_panel component=passportdetails, command=openscript
Note no code

Other simple examples of taggedvalue uses:
- a field’s previous value, held in one of that field’s taggedvalues
- A field’s public name, to use in messages.
... Make data and objects more systematically accessible when needed ...

- Generated userclasses include the database interaction (select, update, insert, delete) as QueryObjects, saving hundreds of lines of SQL code in the frame 4GL.
Provide system methods and templates for currently 4GL-coded features

- Various generic facilities, as well as some specialized but frequently needed facilities, have to be hand-coded by client developers
  - Lists, guidance, validation, database interaction, as well as many smaller-scale features
  - Additionally in 6.2: styling, defined behaviours, animations, ...
- This means delay and development costs (and risk).
- OpenROAD 6.2 provides certain key facilities, in the form of:
  - Helper classes RequestManager, InputEvent, SpriteDescriptor
  - FrameTemplates active_display, table_display, w7style
- Provision of support in these areas is planned to increase further
  - Provided by us
  - Provided by client contributions, for example OpenROAD Sprints
… Provide system templates for currently 4GL-coded features…

➤ The table_display frame template creates frames
  ■ that support simple Data Administration for individual tables,
  ■ without requiring any coding from the developer,
  ■ and are completely standalone
    • no need to create other frames, userclasses or applications
... Provide system methods for 4GL-coded features ...

- The InputEvent and SpriteDescriptor "helper" userclasses contain wrapper methods that make Defined Behaviours, animations and styling easy to implement and maintain
  - Avoiding complex custom 4GL coding
✔ Reduce the cost of OpenROAD development and improve ROI

→ Less 4GL code required, but not more complex or more rigid code

1. Enable removal of setup code from frames
2. Enable storage of predefined behaviours, not coding them at runtime
3. Make data and objects more systematically accessible when needed
4. Provide system methods and templates for currently 4GL-coded features
5. Separate style code from other code

→ Ensure that new features fit in with this

✓
LoadnRun provides a comprehensive and straightforward mechanism for deploying OpenROAD.

Downloadable IngresNet provides a very useful step in redeploying existing Client-Server OpenROAD applications as OpenROAD client-OpenROAD server applications.
III – Improve the partitioning and deployment of OpenROAD

From the customer viewpoint:

1. Provide a comprehensive and straightforward mechanism for deploying OpenROAD
2. Provide richer and more robust support for management of OpenROAD Client-Server and OpenROAD multi-tier architectures
3. Make stepwise redeployment of existing Client-Server OpenROAD applications as OpenROAD multi-tier applications a feasible process
   • Previously this redeployment required all the SQL processing throughout all the client applications to be moved into the Server application before anything could be released
   • Such an all-or-nothing approach if seen as too big and risky a project would block redeployment to multi-tier.
Application deployment using LoadnRun

- LoadnRun provides an infrastructure for OpenROAD deployment
  - Whether the applications are Client-Server or Application Server based
- The aim of the infrastructure is to minimize the deployment effort and risk, and to maximize the range and load that can be handled
  - Central administration - all handled by server, once LoadnRun is installed in the client machines
    - Client installations can be pushed, using Active Directory
  - Deployment of latest versions is automatic
  - Existing client batch processing is readily integrated into LoadnRun
  - Simple-to-use client management tools (where appropriate)
  - The same client machine can run multiple OpenROAD versions (6.2, 6.0, 5.1), multiple installations, multiple applications, multiple instances, multiple users ...
- LoadnRun design was guided by client needs and client experience

Numbers in circles in the title refer to the numbering of the sub-objectives on the previous page
Loadnrun Benefits (already a reality)

- We have already seen practical benefits from the LoadnRun facility:
  - A client has already used OpenROAD 6.2 LoadnRun to deploy and manage their environment:
    - Deployed to 500 PCs
    - 5.1.1 and 6.0.2 OpenROAD applications using the same client machine
    - No problems!
  - Actian use it internally to access and compare older versions of our applications when regression testing
  - Actian use it internally to deploy our own OpenROAD time-management software
Easier partitioning using downloadable Ingres Net

- Direct connection from MultiTier Client to database
  - Instead of / as well as the OpenROAD Server connection
- LoadnRun Launcher uses the local Ingres Net Client configured by LoadnRun
  - The Query Tool provided with the LoadnRun installation includes a vnode-definition wizard that configures the client
✓ Improve the partitioning and deployment of OpenROAD

1. Provide a comprehensive and straightforward mechanism for deploying OpenROAD
2. Provide richer and more robust support for management of OpenROAD Client-Server and OpenROAD MultiTier architectures
3. Make stepwise redeployment of existing Client-Server OpenROAD applications as OpenROAD MultiTier applications a feasible process
Other OpenROAD 6.2 Changes

- Covered in the third of these presentations
- A selected few listed here
  - But there are still more!
And more still ...

- TreeViewField, ControlButton fields
- ToolTipText size
- Curexec system variable
What was covered

- OpenROAD 6.2 – new features addressing customer requirements
  - The first of three presentations covering OpenROAD 6.2

- Remember:
  - Features illustrated in the presentation will require the first OpenROAD 6.2 patch
    - *p14746 or later*
Thank you

- Sean Thrower, OpenROAD Engineering
  sean.thrower@actian.com