OpenROAD 6.2
New Client-Server and Multi-Tier Deployment

New in OpenROAD 6.2 – For all users of OpenROAD.
See: http://community.actian.com/wiki/LoadnRun_Home

Durwin Wright
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.
This presentation contains specific references to Loadnrun 6.2

The information presented is directly applicable to the following

- Loadnrun 5.1
- Loadnrun 6.0

This presentation can be used to support any of the above versions

They only differ in the version of OpenROAD that each supports. Other than that they are virtually identical in behavior.

We recommend the following

- Use Loadnrun 6.2 for the Loadnrun 6.2 Server
- Use Loadnrun 5.1, Loadnrun 6.0 or Loadnrun 6.2 on the clients

Loadnrun 6.2, Loadnrun 6.0 and Loadnrun 5.1 are available now!
OpenROAD 6.2 - New Client-Server and Multi-Tier Deployment

- Features illustrated in the presentation will require the first OpenROAD 6.2 patch
  - p14746 or later
  - Loadnrun 6.2 Installer
  - Loadnrun 6.0 Installer
  - Loadnrun 5.1 installer

- This is the second of three presentations covering OpenROAD 6.2

See: [http://community.actian.com/wiki/LoadnRun_Home](http://community.actian.com/wiki/LoadnRun_Home)
Meeting OpenROAD 6.2 Objectives ...
Improve the deployment of OpenROAD applications

From the customer viewpoint:

- Provide a comprehensive and straightforward mechanism for deploying OpenROAD
- Provide richer and more robust support for management of OpenROAD Client-Server and OpenROAD Multitier architectures
- Make migrations to new versions of applications running the same version of OpenROAD easier
- Make stepwise redeployment of existing Client-Server OpenROAD applications as OpenROAD Multitier applications a feasible process
  - Currently this redeployment requires all the SQL processing throughout all the client applications to be moved into the Server application before anything can be released
  - Clients can view this as too big and risky a project.
Loadnrun provides an infrastructure for OpenROAD deployment

- Client-Server based
- 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
- 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
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 numerous PCs
  - 5.1.1 and 6.0.2 (and testing OR 6.2.0) OpenROAD applications using the same client machine
  - Very few problems

- Actian uses it internally to:
  - To access and compare older versions of our applications when regression testing
  - To test and expose new features under development and compare behaviours of previous versions of OpenROAD side-by-side!
Easier partitioning using optional 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
- The Loadnrun demos show how to use dynamic vnodes that do not require any pre-defined vnodes
- Examples are also provided that show how to create a vnode when the application is first setup after Loadnrun downloads it
Original Loadnrun

- Loadnrun was jointly developed by Actian and Customers

Original Objectives

- Package eClient runtime as an MSI Installer
- Deployment of Loadnrun eClient applications will not require any special privileges for the end user

Customer Contribution

- Loadnrun Client and Server Application
- Visual Studio-based Setup Bootstrapper
- Original documentation on Community Wiki

Actian Contribution

- OpenROAD eClient runtime (based on OR 5.0)
- Originally provided irunnerw.exe (was later deprecated in favor or w4glapp.exe)
- Developed w4glapp.exe (fused version of w4gldev.exe)
New Loadnrun (Part 1 of 3)

- Support for concurrent versions of OpenROAD
  - Loadnrun Client 5.1
  - Loadnrun Client 6.0
  - Loadnrun Client 6.2

- Allow host of any version of OpenROAD eClient by Loadnrun Server
  - Loadnrun Server 5.1 (based on OpenROAD 5.1)
  - Loadnrun Server 6.0 (based on OpenROAD 6.0)
  - Loadnrun Server 6.2 (based on OpenROAD 6.2)

- Packaged optional Standalone Net Client
  - Based on Ingres 10.1.1 Net Client (or 10.0.0 for Loadnrun 5.1/6.0)
  - Hardcoded Installation Codes (XN, XO, XP)
New Loadnrun (Part 2 of 3)

- Packaged as WiX MSI project
- Provided 4GL-based bootstrapper
- Allow Silent or Reduced UI install option
- Support installation into latest versions of Windows with or without UAC enabled
  - Windows 7 and Windows Server 2008 R2
- Incorporated all customer enhancement requests
  - Loadnrun Server Host Isolation
  - Loadnrun Compression
  - PRERUN and POSTRUN scripts
  - Provide feedback during download of user applications
  - Log all Loadnrun Client access to Loadnrun Server
New Loadnrun Objectives (Part 3 of 3)

provide several types of demos
- AppServer-based demos
- Standalone demos
- Two-tier Net Client demos

Provide Simple Launcher
- Demonstrates how to launch a typical application
- This is just a demo but can be incorporated and modified as each customer sees fit

Provide source code for all 4GL applications

Updated Wiki Documentation
http://community.actian.com/wiki/LoadnRun_Home

Add Loadnrun to OpenROAD Documentation set
Getting Started with Loadnrun Client
Loadnrun Terminology

➔ Server

- Loadnrun Gatekeeper
- Loadnrun Server
- Loadnrun Server Runtime
- Loadnrun Server eClient Host Directory

➔ Client

- Loadnrun Client Runtime
- Loadnrun APPNAME
- Loadnrun GATEWAY URL
- Loadnrun SUFFIX
- Loadnrun HOSTNAME
- Loadnrun eClientCache
Loadnrun APPNAME Launching

- The shortcuts for Loadnrun 6.2 are located under the following:
  
  Start ➔ All Programs ➔ Actian Loadnrun 6.2

- The shortcut to launch the Loadnrun Command Window is:
  
  Start ➔ All Programs ➔ Actian Loadnrun 6.2 ➔ Resources ➔ Loadnrun 6.2 Command Window

- These shortcuts will only appear under the Administrative account that was used to install the Loadnrun client runtime:
  
  - The Loadnrun Command, “%II_LOADNRUN62_CMD%”, can be used by any user in a command window or a shortcut
  
  - Essentially, once the Loadnrun Client runtime is installed, the only thing needed on the client machine to deploy and application is the appropriate shortcut
The output of the command, “%II_LOADNRUN62_CMD%”, is shown in the frame below.

This command can be used to launch Loadnrun applications and manage the local cache of OpenROAD Applications.
Start...

...Loadnrun Simple Launcher
...Loadnrun httptest
...Loadnrun connect_example2
...Loadnrun qt
...Loadnrun workbench
Loadnrun Simple Launcher

- Simple application to launch Loadnrun applications
- Sample applications hosted on Loadnrun Server under the 62demo suffix

This can be launched via a shortcut

- Actian Loadnrun 6.2 ➔ Launcher
The Simple Launcher can launch this application

Notice the location URL is configured automatically

Example of AppServer Application

Provided as a demo with Loadnrun
The Simple Launcher can launch this application

This application can be used to generate a dynamic vnode

Example of Two-Tier Client Server application using the S/A Net Client

Example of use of “Dynamic Vnode”

Provided as a demo with Loadnrun
The Simple Launcher can launch this application

This is the Query Tool application

Example of more sophisticated Client/Server application

Provided as a demo with Loadnrun
WorkbenchExpress(62demozip)

This is the Workbench Express application

Example of an even more sophisticated Client/Server application

Not provided as a demo with Loadnrun
Loadnrun Overview
How does Loadnrun Work?

- Loadnrun Server
- Loadnrun Client

- A single MSI-based installer installs and configures the Loadnrun Server and the Loadnrun Client
- Loadnrun Server hosts the User applications and delivers them to the client machines upon request
- Loadnrun Client launches the user application from the Loadnrun Server user application location or the Loadnrun Client local cache
Loadnrun Generic Features

- All versions of the Loadnrun Client can co-exist on the same machine
- Any version of the Loadnrun Client can launch any version of Loadnrun user applications
- Any version of the Loadnrun Server can host any version of the Loadnrun files
- It is possible to install the Loadnrun Client silently or passively
- Once the Loadnrun Client is installed, all that is required to launch an application is the creation of a simple shortcut or use of a command line
- Effectively the user application installer is reduced to a script that can create the appropriate shortcut on the client machine
Loadnrun Server Features

- Each Loadnrun application has an install4gl.txt file and any resources (images and files) that are needed.
- The images and files can be placed in a compressed archive (typically a zip file).
- The images can also be hosted on a network URL.
- A combination of the above is allowed.
- The install4gl.txt file is the only file required to host a user application.

The installation and management of the Loadnrun Server requires Administrative Privileges.
Loadnrun Client Features

- Download of new versions of a Loadnrun applications is automatic
- Version string can be used to force download of new version from server
- Compressed files or network-based image files can be used
- Simple management of applications on local cache
- Simple diagnostic capabilities are built into the product
- Has an optional Ingres Net Client for client/server applications
- Applications from different Loadnrun URLs do not share the same eclientcache sub-directory
- Control can be given to a user written script before the Loadnrun 4GL application is launched
- Does not require any special privileges to download, launch and run a Loadnrun application

Installation of Loadnrun Client Runtime does require Administrative privileges
Launching Application: Cache Hit on Client
# Loadnrun Launch – Use Local eclient cache version

<table>
<thead>
<tr>
<th></th>
<th>Loadnrun</th>
<th>eclient</th>
<th>Loadnrun</th>
<th>eclient</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>Client</td>
<td>Cache</td>
<td>Server</td>
<td>Host</td>
</tr>
</tbody>
</table>
Loadnrun Launch – Use Local eclient cache version

<table>
<thead>
<tr>
<th>User</th>
<th>Loadnrun Client</th>
<th>eclient Cache</th>
<th>Loadnrun Server</th>
<th>eclient Host</th>
</tr>
</thead>
</table>

Launch App1
Loadnrun Launch – Use Local eclient cache version

<table>
<thead>
<tr>
<th>User</th>
<th>Loadnrun Client</th>
<th>eclient Cache</th>
<th>Loadnrun Server</th>
<th>eclient Host</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Launch App1</td>
<td>Get user_version</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Launch App1
- Get user_version
Loadnrun Launch – Use Local eclient cache version

User -> Loadnrun Client -> eclient Cache -> Loadnrun Server -> eclient Host

- Launch App1
- Get user_version
- checkFiles()
Loadnrun Launch – Use Local eclient cache version

User → Loadnrun Client → eclient Cache → Loadnrun Server → eclient Host

- Launch App1
- Get user_version
- checkFiles()
- Check server_version
Loadnrun Launch – Use Local eclient cache version

User: Launch App1
Loadnrun Client: Get user_version
eclient Cache: checkFiles()
Loadnrun Server: Check server_version
eclient Host: Use eclient cache version
Loadnrun Launch – Use Local eclient cache version

User

Launch App1

Loadnrun Client

Get user_version

checkFiles()

Launch eclient cache App1

eclient Cache

Loadnrun Server

Check server_version

eclient Host

Use eclient cache version
Loadnrun Launch – Use Local eclient cache version

User

Launch App1

Loadnrun Client

Get user_version

checkFiles()

eclient Cache

Launch eclient cache App1

Loadnrun Server

Check server_version

Use eclient cache version

eclient Host

App1 Launched

App1 Launched
Launching Application: Cache Miss on Client
Loadnrun Launch – Get eclient Host version
Loadnrun Launch – Get eclient Host version

User    | Loadnrun Client | eclient Cache | Loadnrun Server | eclient Host

Launch App1
Loadnrun Launch – Get eclient Host version

User → Loadnrun Client → eclient Cache → Loadnrun Server → eclient Host

- Launch App1
- Get user_version
Loadnrun Launch – Get eclient Host version

User → Loadnrun Client → eclient Cache → Loadnrun Server → eclient Host

- Launch App1
- Get user_version
- checkFiles()
Loadnrun Launch – Get eclient Host version

User → Loadnrun Client → eclient Cache → Loadnrun Server → eclient Host

1. Launch App1
2. Get user_version
3. checkFiles()
4. Check server_version

Loadnrun Launch – Get eclient Host version
Loadnrun Launch – Get eclient Host version

User → Loadnrun Client → eclient Cache → Loadnrun Server → eclient Host

Launch App1

Get user_version

checkFiles()

Use eclient Host version

Check server_version
Loadnrun Launch – Get eclient Host version

Launch App1

User → Loadnrun Client → eclient Cache → Loadnrun Server → eclient Host

Get user_version

checkFiles()

Use eclient Host version

Check server_version

getFiles()
Loadnrun Launch – Get eclient Host version

User

Loadnrun Client

Get eclient Host version

User

Loadnrun Client

Launch App1

Get user_version

checkFiles()

getFiles()

Loadnrun Cache

Loadnrun Server

Check server_version

Use eclient Host version

Loadnrun Server

Fetch eclient Host Files

eclient Host

Loadnrun Server

Check server_version

Use eclient Host version

Loadnrun Server

Fetch eclient Host Files

eclient Host

Loadnrun Cache

Get user_version

checkFiles()

getFiles()
Loadnrun Launch – Get eclient Host version

<table>
<thead>
<tr>
<th>User</th>
<th>Loadnrun Client</th>
<th>eclient Cache</th>
<th>Loadnrun Server</th>
<th>eclient Host</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Loadnrun Client</td>
<td>eclient Cache</td>
<td>Loadnrun Server</td>
<td>eclient Host</td>
</tr>
<tr>
<td></td>
<td>Launch App1</td>
<td>Get user_version</td>
<td>checkFiles()</td>
<td>Check server_version</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>getFiles()</td>
<td>Use eclient Host version</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Fetch eclient Host Files</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Return eclient Host files</td>
</tr>
</tbody>
</table>
Loadnrun Launch – Get eclient Host version

User

Loadnrun Client

eclient Cache

Loadnrun Server

eclient Host

Launch App1

Get user_version

checkFiles()

getFiles()

Create eclient cache App1

Check server_version

Use eclient Host version

Fetch eclient Host Files

Return eclient host files
Loadnrun Launch – Get eclient Host version

User
Loadnrun Client
eclient Cache

Launch App1
Get user_version
checkFiles()
getFiles()
Create eclient cache App1
Launch eclient cache App1

Loadnrun Server
eclient Host

Check server_version
Use eclient Host version
Return eclient Host files
Fetch eclient Host Files

Create eclient cache App1
Launch eclient cache App1
Get user_version
checkFiles()
getFiles()
Loadnrun Launch – Get eclient Host version

User

Loadnrun Client

eclient Cache

Loadnrun Server

eclient Host

Launch App1

Get user_version

checkFiles()

getFiles()

Create eclient cache App1

Launch eclient cache App1

Check server_version

Use eclient Host version

Fetch eclient Host Files

Return eclient Host files

App1 Lunched

getFiles()

Create eclient cache App1

Launch eclient cache App1

Return eclient host files
Demo ...
Loadnrun: Multiple Versions
Multiple Versions of Loadnrun

- Each Loadnrun runtime can
  - Access the Loadnrun Server
  - Download a hosted application if it exists
  - Launch the application using the correct runtime

- In order for a Loadnrun 6.2 runtime to launch a Loadnrun 5.1 or Loadnrun 6.0
  - The application must exist on the Loadnrun Server under the eClient sub-directory
  - The appropriate Loadnrun runtime must exist on the Client machine.

- This also means that OpenROAD 5.1, OpenROAD 6.0 and OpenROAD 6.2 applications and test and production versions can co-exist on the same client machine and be launched and run independent of each other

- This behavior is helpful in migrations from older OpenROAD versions to newer OpenROAD versions (for example OR 5.1 to OR 6.2)
The Loadnrun Server hosts the files under the following directory:

- \%II_LOADNRUN62_W4GLAPPS_DIR%\eClient

This directory contains sub-directories that correspond to the Loadnrun Suffix value:

- 62demo
- 62prod
- 62test

Note that the value of this suffix does not have anything to do with the version of the OpenROAD applications.
The subdirectories in this location is as follows

- 51demo
- 51prod
- 51test
- 60demo
- 60prod
- 60test
- 62demo
- 62prod
- 62test

Notice that Loadnrun 6.2 is hosting applications from other versions of OpenROAD
"%II_LOADNRUN62_CMD%" /listserver 62demo
"%II_LOADNRUN62_CMD%" /listserver 51demo
"%II_LOADNRUN62_CMD%" /listserver 60demo
"%II_LOADNRUN51_CMD%" launcher %II_LOADNRUN51_GATEWAY% 51demo

<table>
<thead>
<tr>
<th>HTTP Test Demo (51demo)</th>
<th>Comtest Client demo (51demo)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jigfall Demo (51demo)</td>
<td>Minesweeper Demo (51demo)</td>
</tr>
<tr>
<td>Intertask1 (51demo)</td>
<td>Intertask2 (51demo)</td>
</tr>
<tr>
<td>connect_example2 (51demo)</td>
<td>Query Tool (51demo)</td>
</tr>
</tbody>
</table>

Exit
"%II_LOADNRUN51_CMD%" launcher %II_LOADNRUN51_GATEWAY% 60demo
"%II_LOADNRUN51_CMD%" launcher %II_LOADNRUN51_GATEWAY% 62demo
Configure the IIS Gatekeeper
The OpenROAD Gatekeeper needs to be configured for Loadnrun.

- This provides HTTP/HTTPS access to the OpenROAD Server
- Uses a Web Server application

There are three versions of the OpenROAD Gatekeeper:

- IIS ASP.NET version (IIS 6 or IIS 7.x)
- Tomcat 32-bit Java version
- Tomcat 64-bit Java version

In this section the IIS 7.0 ASP.NET version is used

All versions have been configured using the existing OpenROAD procedures and documentation
The application can be launched via the URL

- When the URL for the gatekeeper is passed, then it is launched
- The machine name is passed explicitly rather than localhost
The comtestrso command can be used to verify access to the OpenROAD Server via the Gatekeeper.

This test verifies that requests can be made to the OpenROAD Server via an HTTP request.
OpenROAD Gatekeeper is now configured

- The platforms that have been tested are
  - Windows 7 (32-bit/64-bit)
  - Windows 8 (32-bit/64-bit), Windows 8.1 (32-bit/64-bit)
  - Windows Server 2008 R2 (64-bit)
  - Windows Server 2012 (64-bit)

- OpenROAD and Loadnrun Versions tested are
  - OpenROAD 5.1.1+, Loadnrun 5.1
  - OpenROAD 6.0.2+, Loadnrun 6.0
  - OpenROAD 6.2.0, Loadnrun 6.2
Configure the Tomcat 64-bit Gatekeeper
OpenROAD Gatekeeper

→ The OpenROAD Gatekeeper needs to be configured for Loadnrun.
  • This provide HTTP/HTTPS access to the OpenROAD Server
  • Uses a Web Server application

→ There are three versions of the OpenROAD Gatekeeper
  • IIS ASP.NET version (IIS 6 or IIS 7.x)
  • Tomcat 32-bit Java version
  • Tomcat 64-bit Java version

→ In this section the Tomcat 64-bit Java version is used

→ All versions have been configured using the existing OpenROAD procedures and documentation
The `%II_SYSTEM%\INGRES\bin64` directory (1 or 2)

- Can be used to access the OpenROAD server (32-bit) from 64-bit processes (e.g. a JVM) on Windows 64-bit installations.

- The files contained in this directory are:
  - ororso.dll
  - orps.dll
  - orrsojni.dll
  - comtestrso.exe

- These are 64-bit versions of those delivered in `%II_SYSTEM%\INGRES\bin`. 
The %II_SYSTEM%\inges\bin64 directory (2 or 2)

→ The Microsoft Visual C++ runtime redistributables (64-bit) are
  ▪ mfc100.dll
  ▪ mfc100u.dll
  ▪ mfcm100.dll
  ▪ mfcm100u.dll
  ▪ msvcp100.dll
  ▪ msvcr100.dll

→ This runtime is needed to support the 64-bit OpenROAD Deliverables.
Configuring the 64-bit OpenROAD Deliverables for use with Tomcat

- Use the files contained in the %II_SYSTEM%\ingres\bin64 directory

- Register the bin64-DLLs orrso.dll and orps.dll using %windir%\system32\regsvr32 (64-bit version)
  - %windir%\system32\regsvr32 %II_SYSTEM%\ingres\bin64\orps.dll
  - %windir%\system32\regsvr32 %II_SYSTEM%\ingres\bin64\orrso.dll

- Add %II_SYSTEM%\ingres\bin64 directory in front of the PATH (or other) environment variable(s) where required (when using 64-bit processes)
  - For Tomcat configure its “Java” settings – add the Java option:
  - -Djava.library.path=<full path of %II_SYSTEM%\ingres\bin64>
The application can be launched via the URL

- When the URL for the gatekeeper is passed, then it is launched
- The machine name is passed explicitly rather than localhost
The comtestrso command can be used to verify access to the OpenROAD Server via the Gatekeeper.

This test verifies that requests can be made to the OpenROAD Server via an HTTP request.
Install and Configure Loadnrun
Install Loadnrun

- Launch the Loadnrun 6.2 installer
- Enter information as shown in the frame to the right
- This options will configure the Loadnrun Server and the Loadnrun Client on the same machine
- The Server Suffix is used to specify the default location of user applications
Accept of modify S/A Net Options

- The Ingres 10.1.1 client has been selected for installation
- The Installation Code cannot be modified but the other options can be modified
Register Loadnrun to the OpenROAD Server
Verify Environment Variables for Loadnrun

- Verify that the II_LOADNRUN62 environment variables are set
- Verify that the II_W4GLAPPS_SYS is correct
- Verify that the II_W4GLAPPS_DIR is correct and points to the same location as II_LOADNRUN62_W4GLAPPS_DIR
- (The system may need to be rebooted if II_W4GLAPPS_DIR was changed from a previous value in order for the OpenROAD Server to pick it up.)
Use VOSA to configure Loadnrun

Launch VOSA to configure Loadnrun
Define Loadnrun to the OpenROAD Server

➢ Use the Register option to define Loadnrun
Initial Register Frame

The frame to the right appears when starting the Register operation for Loadnrun
It is assumed in this example that II_W4GLAPPS_DIR points to the location that has the loadnrun.img file.
Save the Loadnrun Registration

When the Loadnrun registration information is saved, the frame to the right will appear.
Verify Loadnrun Application is Registered

- VOSA should be able to access and display the Loadnrun application SCPs
Use comtestrso to access Loadnrun

The comtestrso application can be used to access Loadnrun via the Gatekeeper.
Conclusion

OpenROAD 6.2 - New Client-Server and Multi-Tier Deployment

- Features illustrated in the presentation will require the first OpenROAD 6.2 patch
  - p14746 or later
  - Loadnrun 6.2 Installer
  - Loadnrun 6.0 Installer
  - Loadnrun 5.1 installer

- This was the second of three presentations covering OpenROAD 6.2

See: [http://community.actian.com/wiki/LoadnRun_Home](http://community.actian.com/wiki/LoadnRun_Home)
Thank you

Durwin Wright
OpenROAD Engineering
durwin.wright@actian.com
http://community.actian.com/wiki/LoadnRun_Home