

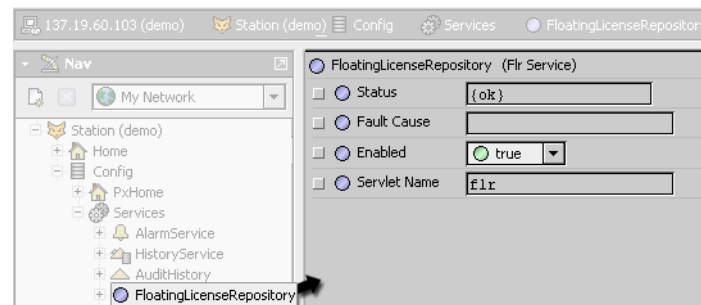
Information and/or specifications published here are current as of the date of publication of this document. Tridium, Inc. reserves the right to change or modify specifications without prior notice. The latest product specifications can be found by contacting our corporate headquarters, Richmond, Virginia. Products or features contained herein are covered by one or more U.S. or foreign patents. This document may be copied by parties who are authorized to distribute Tridium products in connection with distribution of those products, subject to the contracts that authorize such distribution. It may not otherwise, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form without prior written consent from Tridium, Inc. Complete Confidentiality, Trademark, Copyright and Patent notifications can be found at: <http://www.tridium.com/galleries/SignUp/Confidentiality.pdf>. © 2010 Tridium, Inc.

JACE, Niagara Framework, Niagara AX Framework and the Sedona Framework are trademarks of Tridium, Inc.

NiagaraAX Floating License Repository (FLR)

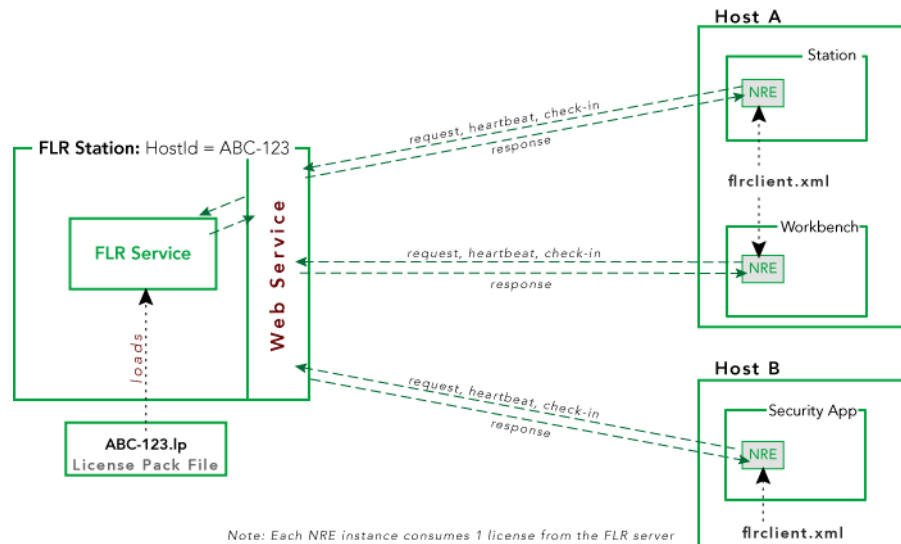
The FLR is a licensed feature that runs as a service in a Niagara station to provide licenses to other Niagara Runtime Environment (NRE) applications.

Figure 3-1 FLR service and property sheet view



In this environment, available licenses are “leased” by authorized NRE applications at startup and “returned” when the application shuts down. [Figure 3-2](#) shows the basic FLR environment architecture. In the illustration, below, three licenses are being consumed by the two Hosts (Host A, Host B).

Figure 3-2 FLR environment architecture



The FLR environment is comprised of the following:

- FLR server**
 The Floating License Repository (FLR) is a NiagaraAX service that runs in a user's network environment to provide host-independent licensing for a limited number of NRE instances (FLR clients). The FLR server uses an LPF to provide licensing to properly configured FLR clients.

- **FLR client**

Niagara applications run in the Niagara Runtime Environment (NRE) and include one or more instances of a Niagara station, workbench, driver, service, or other application. An FLR client is any NRE application *instance* that is configured to get its license from an FLR server.

Each client that is licensed by an FLR server “leases” one of the FLR server’s available licenses. If no licenses are available, the application will not start. When the NRE application instance closes, the license is “returned” to the FLR server and is available for any properly configured FLR client.

Note: *The concept of “instance” is very important. Each floating NRE application impacts the number of free licenses on the FLR.*

When the FLR starts running, it reads the LPF and starts accepting FLR client requests for licenses. The FLR maintains the state of each license in the pack and responds to heartbeat messages from NREs that have a leased license. See also, “[About floating licenses](#)” on page 3-2 and “[About the license pack file \(LPF\)](#)” on page 3-3. “[Example FLR scenarios](#)” on page 3-5.

Note: *The system clocks on the FLR client and server must be within one minute of each other. If the system clocks differ by more than a minute, the client is not able to check out a license from the server. For this reason, it may be helpful to set up a time synch service for the server and clients. See the section [About the NtpPlatformService](#) in the NiagaraAX Platform Guide for more information about setting up a synchronized time service.*

The following sections describe the primary characteristics of and procedures relating to the Floating License Repository (FLR):

- [About node-locked licenses](#)
- [About floating licenses](#)
- [About the license pack file \(LPF\)](#)
- [About the flrclient.xml file](#)
- [FLR Configuration procedures](#)
- [Example FLR scenarios](#)
- [Document change log](#)

About node-locked licenses

Niagara licensing in versions prior to NiagaraAX-3.5, uses a “node-locked” model, only. This means that every NiagaraAX license that is generated is unique for a specific host and will not work with any other host. This model is enforced by generating a license that is based on a unique and unalterable value (**hostID**) that is assigned to the hosting hardware.

Note: *Node-locked licensing is still implemented and available in NiagaraAX-3.5. A station running an FLR service requires a node-locked license to start.*

About floating licenses

Starting with NiagaraAX-3.5, any time a Niagara Runtime Environment (NRE) application boots, it checks the installation “licenses” directory for an `flrclient.xml` file. This is a file that tells the NRE where to find a “floating license” (the address and port of an FLR server).

Note: *The “floating license” uses a `hostId` value of “FLOATING” instead of a unique `hostID`.*

If the `flrclient.xml` file is present and valid, the NRE tries to get a license from one of the FLRs defined in the file. If the file is present but not valid, or if the file is not present at all, the NRE attempts to load a local (node-locked) license. The NRE instance does not start unless it is licensed.

After the NRE application instance successfully starts using a floating license, the license manager continues to send periodic heartbeat messages to the FLR to maintain its lease on the floating license. As long as this heartbeat mechanism is working, the application continues to run. If heartbeats start failing, a retry-loop is entered and the application eventually is forced to stop unless it starts getting heartbeats again.

When the application stops, the license manager indicates to the FLR that it no longer needs the license so that another application can start using that license, if needed. In the case of an ungraceful stop, the FLR eventually auto-releases the license after a set period of failed heartbeats.

Also see “[Example FLR scenarios](#)” on page 3-5.

About the license pack file (LPF)

LPFs are created on the Niagara License Server as part of the license record for the FLR. Note the following points about the LPE.

LPF characteristics

- The name of the file is fixed and bound to the hostId of the system. The file name format for an LPF is as follows: <hostId>.lp
- The LPF is a “signed” file that you should NOT edit.
- The FLR always starts running if it has a valid node-locked license but it does not serve licenses if the LPF is not valid.
- Any updated licensing (for example, to change a port number for the FLR) requires the generation of a new LPF. When the new license (with the same name) is installed, the previous LPF is overwritten by the new license. While running, the FLR periodically checks the LPF to make sure that it has not changed. If a change is detected, the FLR attempts to reload the new (changed) file. If the port has changed, the FLR restarts itself on the new port. This prevents multiple FLRs from running on the same host.
- The FLR allows the LPF to be reloaded without requiring a restart.

Refer to [Figure 3-3](#) and the following list for descriptions each of the major blocks of the LPF:

Figure 3-3 Example LPF

```
<lpf vendor="Acme" generated="2010-01-31" expiration="2011-01-31">
  <!-- This is the configuration for the FLR -->
  <flr hostId="hostId" port="19110" />
  <!-- These are the license packs that the FLR can serve -->
  <packs>
    <!-- A "station" pack -->
    <pack name="station" limit="10">
      <licenses>
        <!-- This is a standard licenses block at this point. All the
        licenses that are in here will be sent to NREs that request the "station" pack.
        -->
      </licenses>
    </pack>
    <!-- A "workbench" pack -->
    <pack name="wb" limit="20">
      <licenses>
        <!-- ... -->
      </licenses>
    </pack>
    <!-- A "energy" pack -->
    <pack name="energy" limit="2">
      <licenses>...</licenses>
    </pack >
  </packs>
  <signature>signed by Licensor</signature>
</lpf>
```

LPF elements and attributes

- **<lpf> element**
This element is the top level element for the LPF. It includes attributes for defining the “vendor” name, file “generated” date, and an “expiration” date for the LPF.
- **<flr> element**
This element contains the configuration for the FLR that it uses when it starts. The “hostId” attribute value is necessary to prevent starting FLRs on unauthorized hosts. The “port” attribute is specified to prevent multiple copies from running on the same host.
If you ever want to change the port that the FLR is running on, a new LPF must be generated and downloaded from Niagara License Server (or sent by email).
- **<packs> element**
This element is the parent for one or more individual license packs that can be served by the FLR. See <pack> element, below.
- **<pack> element**
All <pack> elements are uniquely named packs, using the “name” attribute. These may be anything; they are simply identifiers for license configurations. The content of a <pack> element is the same content as the traditional NiagaraAX license file. For example, an LPF may have three packs: a “sta-

tion" pack a "workbench" pack, and an "energy" pack. The floating NREs use the pack name to request a license from the FLR.

All <pack> elements have a "limit" attribute that specifies the maximum number of application instances that can use this pack. For example, consider the "station" pack in the LPF in [Figure 3-3](#). It has a "limit" attribute of 10 which means that only 10 NREs can have this pack checked out at any given time. Inside each pack is a single <licenses> XML element.

- **<licenses> element**

The <licenses> block is basically a standard Niagara license with a few modifications. All the licenses in this block (and their features) are sent to a floating NRE when it leases the "station" pack. There is only one set of licenses defined in each pack. There is not a limited number of licenses in a pack. This means that each floating NRE that requests the "station" pack gets the same license. However, this license is never written to disk and does not work as a stand-alone license.

Refer to the *NiagaraAX User Guide* for more details about the standard NiagaraAX license file.

About the flrclient.xml file

An flrclient_template.xml file is provided in the "licenses" folder as part of the NiagaraAX Supervisor default installation. In order to retrieve a license, you need to rename this template file to flrclient.xml and edit the <flr> and <packmap> attributes as described below. The flrclient.xml file provides the information that the NRE license manager requires to locate the FLR and a set of application-to-license-pack mappings that the NRE uses to determine which license it should lease from the FLR.

Figure 3-4 flrclient.xml file

```
<flrclient>
  <flr host="100.123.3.3" port="1234" />
  <flr host="myhost.acme.org" port="1234" />
  <!-- convenience for app="com.tridium.station.Station" -->
  <packmap app="station" pack="myStationPack" />
  <!-- convenience for app="com.tridium.workbench.shell.WbMain" -->
  <packmap app="wb" pack="wbPack" />
  <!-- some other licensed application -->
  <packmap app="com.tridium.foo.Bar" pack="barPack" />
</flrclient>
```

Note: When starting up an NRE application, the NRE license manager looks for a valid flrclient.xml file. If a valid file is not found, the license manager looks next for a standard node-locked license.

The following list includes brief descriptions of the required elements in the flrclient.xml file:

- **<flrclient>**
This top level element contains all other elements in the file.
- **<flr>**
Use this element's attributes to define the FLR server address and port, as described below:
Note: The flrclient.xml file may contain multiple FLR configurations (specified by multiple <flr> elements) so that you can distribute license request loads, if desired. Also, you can edit the flrclient.xml file at any time to add, remove, or update flr configurations.
 - **host**
Type in the ip address of the FLR server (see [Figure 3-4](#))
 - **port**
Type in the FLR port number. This number is defined in the FLR server's license pack file (see [Figure 3-4](#) and "About the license pack file (LPF)" on page 3-3).
- **<packmap>**
Use this element to identify the NRE application and complementary license pack name
 - **app**
Type in the name of the app (see [Figure 3-4](#)). Note that the app names for Workbench and Station allow a convenience abbreviation "wb" and "station", respectively. Other applications require the full java class name.
 - **pack**
Type in the name of the app, as defined in the license pack file. The value for this attribute is specific to your organization.

Example FLR scenarios

The following example scenarios are provided to help explain the typical FLR runtime and licensing process:

- “Runtime scenarios” on page 3-5
- “Licensing scenarios” on page 3-6

Runtime scenarios

A “heartbeat” communication between an NRE with a floating license and the FLR server is used to enforce the licensing limits for a particular license bundle.

Note: *The following examples use a “station” as the NRE application, but the behavior applies to any NRE instance using the FLR.*

Consider the following example environment:

1. An FLR server has a single license pack with:
 - name="station"
 - limit="1" (only one station can get the license at a time).
2. There are two supervisor stations (“S1” and “S2”) configured to use the FLR to get a license.
3. S1 starts and acquires the "station" license from the FLR.
4. S2 is not running.
5. S1, running normally, sends periodic heartbeat messages to the FLR to maintain its possession of the license.

Given the above FLR environment state, consider the following possible scenarios:

- **Scenario 1: S2 starts**
Since S1 has obtained the only license from the FLR, the S2 request is rejected and the station does not start. S2 provides an error message stating that it is unable to obtain a license.
- **Scenario 2: S1 stops normally**
The NRE on S1 notifies the FLR that it is finished with the license, thus freeing up a license for another station to start. An authorized station can now acquire the single "station" license from the FLR server.
- **Scenario 3: The FLR server is shutdown or “killed”**
The next heartbeat from S1 fails because of no response from the FLR server. S1 then goes into a retry state. There are two possible outcomes:
 - **The FLR comes back up before the retry period ends.**
In this case, the next heartbeat from S1 succeeds and S1 attempts to reacquire its license from the FLR. If S1 reacquires the license it continues running normally. If it fails to reacquire the license (for example, the license is no longer available), it shuts down.
 - **The FLR does not come up before the retry period ends.**
In this case, S1 will become unusable and shut down.
- **Scenario 4: FLR is restarted, then S2 is started**
This scenario is similar to Scenario 3, except now a competition, or “race” condition is introduced. Since the FLR was restarted, it has lost all its state information (license counters, etc.). When it starts up it assumes all licenses are available to serve. There are two possible results:
 - **S1 heartbeat gets to the FLR before S2 attempts to get a license:**
In this case, S1 indicates to the FLR that it currently has the license. This causes the FLR to reduce the number of available licenses (to 0 in this case). When S2 attempts to get a license, it fails since there are no more licenses.
 - **S2 requests a license before the next S1 heartbeat goes to the FLR:**
In this case, S2 acquires the license since the FLR thinks it has one to serve. When S1 next sends a heartbeat to the FLR, the FLR rejects the request from the NRE and the NRE immediately goes into an unusable state.
- **Scenario 5: S1 is stopped ungracefully (killed), then S2 is started**
In this scenario, S1 was “killed” and did not have a chance to release its license to the FLR. There is nothing the FLR can do about this since it does not actively ping every NRE it has served a license to. In this case, when S2 starts it does not get a license since the FLR shows that S1 is using it. The FLR maintains an internal timeout and expiration for the license it served to S1. After a predetermined amount of time it auto-releases the lock for that license if no heartbeat is received from S1. In this case, if you do not want to wait for the timeout period to expire, you could restart the FLR. Restarting causes all licensed floating NREs to compete to reacquire a license on the next heartbeat. This introduces the “race” condition, similar to the situation in Scenario 4.

Licensing scenarios

• Scenario 1 – License a new FLR

Request a license from a licensing provider and provide the following information:

- The hostId of the platform that runs the FLR server application
- The port that the FLR should listen on
- The desired license pack configurations for the FLR to serve

The Niagara Licensing Server generates the following files for the FLR

- Node-locked license file for the FLR. This license has the FLR feature enabled in it. The FLR cannot start without this license.
- The signed license pack file (LPF) for that FLR. The FLR can run without an LPF file, but it does nothing in that case. It needs to be told to reload the LPF file if there is an update or error with that file.

• Scenario 2 – Change the port for an FLR

The LPF file needs to be regenerated and sent to the customer. There are two mechanisms that prevent running two FLRs are the same host in this case.

- **The LPF file has a fixed name, so the new one must overwrite the previous one.**
- **The FLR periodically scans the LPF file to make sure it hasn't changed.**

If it detects a change, it stops serving licenses, and all proxy NREs eventually fail because the heartbeat is not getting a response.

Note: When an FLR changes its port, all the client NREs that are configured for that FLR must be reconfigured to use the new port.

• Scenario 4 – Change a license pack or feature set

The LPF file needs to be regenerated by the Niagara License Server and sent to the customer. The FLR can either be restarted or have the LPF file "reload". If a reload is performed and the pack limit becomes smaller than the current value some NREs will be rejected and immediately fail.

If a feature set changes for a pack, you probably will want to reload the license on all client NREs that already have that pack leased. However, it is not guaranteed that applications in the running NRE will "see" the feature set change because many applications only check their features once, and cache whether it is available or not. In this case, the client application would need to restart in some cases and reacquire the license to see the new set of features

FLR Configuration procedures

This section includes tasks that describe how to:

- [Configure an FLR server](#)
- [Configure an FLR Client](#)
- [Check license packs](#)
- [Reload the License Pack File \(LPF\)](#)

Configure an FLR server

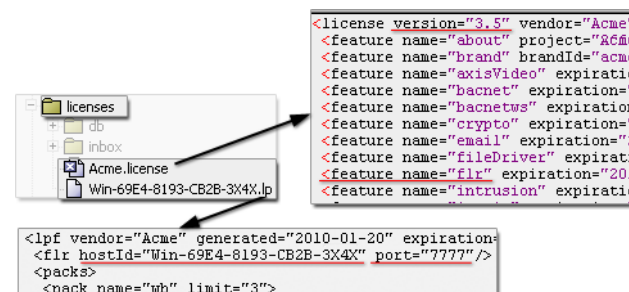
This procedure describes how to use WorkbenchAX to configure an FLR server application. Before configuring the FLR server, check that the following licensing and station status prerequisites are met (see [Figure 3-5](#)):

• Licensing prerequisites:

• node-locked license

The FLR server station must run on a host platform using a node-locked (fixed hostID) license that has a minimal set of features to support running the FLR service.

Figure 3-5 License directory with node-locked license and LPF

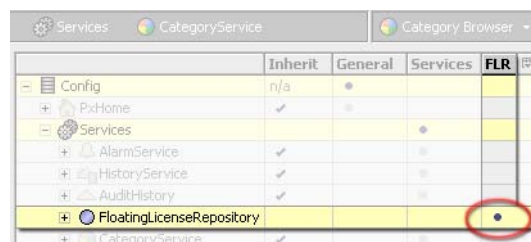


- **“flr” feature**
The node-locked license file must contain the “flr” feature.
- **version 3.5 or later**
The node-locked license must be licensed for version 3.5 or later
- **license pack file**
The FLR requires a signed license pack file (LPF). This file is available from the Niagara License Server and should be located in the “license” folder. The LPF specifies configuration information (hostID and port number) for the FLR as well as the actual license packs.
Note: The FLR can start running if it has a valid node-locked license, but it will not serve licenses if the license pack file is not valid.
- **Station prerequisites**
 - **Station running**
The FLR server station must be running
 - **WorkbenchAX to station connection**
You must establish a Workbench AX connection to the FLR server station (Logon credentials required).

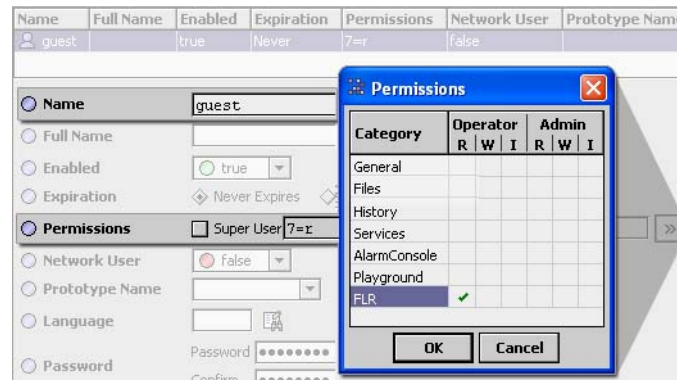
To configure the FLR server, do the following:

- Step 1 In WorkbenchAX, use the Palette sidebar and the **Open Palette** dialog box to open the flr Palette. The flr palette displays in the Palette sidebar.
- Step 2 Drag and drop the FloatingLicenseRepository component from the flr Palette onto the Services folder in the nav tree.
The FloatingLicenseRepository component displays under the Services node in the nav tree.
- Step 3 In the nav tree, right-click on the CategoryService component and select **Views > Category Manager** from the popup menu.
The Category Manager view displays.
- Step 4 In the Category Manager view, click the **New** button at the bottom of the view.
The **New** dialog box displays.
- Step 5 In the **New** dialog box click the **OK** button to add a single new category.
An expanded **New** dialog box displays.
- Step 6 In the expanded **New** dialog box, type “FLR” in the name field for the new category (“FLR” is recommended, not required) and click the **OK** button.
The new category appears in the Category Manager table listing.
- Step 7 In the nav tree, right-click on the CategoryService component and select **Views > Category Browser** from the popup menu.
The Category Browser view displays.
- Step 8 In the Category Browser view, expand the **Config** and **Services** nodes and click under the FLR category column to add only the FLR category to the FloatingLicenseRepository row, as shown below.

Figure 3-6 Adding the FloatingLicenseRepository to the FLR category



- Step 9 In the nav tree, right-click on the UserService node and select **Views > User Manager** from the popup menu. The User Manager view displays.
- Step 10 In the User Manager view, double-click on the “guest” account under the Name column. The Edit dialog box displays.
- Step 11 In the **Edit** dialog box, at the Permissions property, click the >> button, to open the **Permissions** dialog box and set the guest account to have only Operator Read permission, as shown below.

Figure 3-7 Adding read permission to the “guest” account

Step 12 Click the **OK** buttons to close all open dialog boxes.

The FLR server is configured.

Note: You can verify that the FLR service is running by typing `http://localhost:<port>flr/packs` in a browser on the FLR platform.

Configure an FLR Client

This procedure describes how to configure an FLR client application. This applies to NiagaraAX-3.5 or later installations.

Note: The `flrclient_template.xml` is installed by default in the license directory of any NiagaraAX-3.5 Supervisor installation.

Step 1 On the client platform, make a copy of the `flrclient_template.xml` file in the “licenses” directory and name it `flrclient.xml` (or just rename the original `flrclient_template.xml` file).

Note: The `flrclient_template.xml` file contains a helpfully commented example configuration.

Step 2 Edit the `flrclient.xml` file to have at least one `<flr>` element.

Step 3 In the `flrclient.xml` file, edit the workbench `<packmap>` element so that it specifies the name of your actual license application and license pack. For example, `<packmap app="wb" pack="<packName>" />` and save the file.

Note: The “wb” value for “app” is an example for starting a workbench application. The value “station” is used for starting a station. Other values require the full java class name. The value for the “pack” attribute is specific to your organization.

Step 4 Start the application on the client host.

If you are using the console mode (or console window) to launch the application, you can view the startup output.

Following, is an example of typical console output log data for a successfully licensed application:

```
MESSAGE [11:04:56 08-Sep-09 EDT][sys.license] Obtained license pack
"workbenchPack" from FLR http://localhost:7777/flr/
```

Note: The following list of error messages is not an exhaustive list. Many more error messages are possible.

Example error messages

- **If the FLR is not available, you see an ERROR message like this in the console log:**

```
ERROR [11:10:45 08-Sep-09 EDT][sys.license] Failed to lease license pack
"workbenchPack" from FLR http://localhost:7777/flr/. -- Message to FLR failed:
java.net.ConnectException: Connection refused: connect
```
- **If you have the wrong “pack name” in the `flrclient.xml` file, or if the FLR does not serve that license pack, you see an ERROR message like the following in the console log:**

```
ERROR [11:13:06 08-Sep-09 EDT][sys.license] Failed to lease license pack
"workbenchPack" from FLR http://localhost:7777/flr/. --
com.tridium.sys.license.FlException: Pack 'workbenchPack' is not defined in
this FLR.
```
- **If the FLR license limit has been reached for a particular license pack, you see an ERROR message like this in the console log:**

```
ERROR [11:15:12 08-Sep-09 EDT][sys.license] Failed to lease license pack
"workbenchPack" from FLR http://localhost:7777/flr/. --
com.tridium.sys.license.FlException: The license limit has been reached for
pack [workbenchPack]. limit = {2}.
```


- **If there are any errors (whatever the type), workbench fails to start and you get an error dialog with the following stack trace details**

```

javax.baja.license.LicenseDatabaseException: Could not lease license pack
"workbenchPack" from any configured FLR at com.tridium.sys.license.NLicenseM-
anager.checkFeature(NLicenseManager.java:99)
    at com.tridium.workbench.shell.WbMain.checkLicense(WbMain.java:528)
    at com.tridium.workbench.shell.WbMain.checkWorkbenchLi-
cense(WbMain.java:506)
    at com.tridium.workbench.shell.WbMain.main(WbMain.java:755)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(Unknown Source)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source)
    at java.lang.reflect.Method.invoke(Unknown Source)
    at com.tridium.sys.Nre.runClass(Nre.java:201)
    at com.tridium.sys.Nre.main(Nre.java:126)

```

Check license packs

You can verify that an FLR service is running and check license pack availability and status at the FLR server by doing the following:

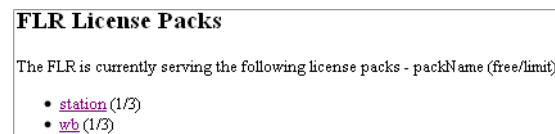
- Step 1 Open a browser session on the FLR server platform and type in the following:

`http://localhost:nnnn/flr/packs`

where “nnnn” is the port that the FLR is configured to listen on.

If the FLR service is running, the FLR License Packs page displays the number of “in use” license packs and the number of total license packs, by pack, as shown below:

Figure 3-8 FLR License Packs page



- Step 2 In the FLR License Packs view, click on the “station” link (or any pack name link that displays).

The License Pack Status page displays, as shown below:

Figure 3-9 License Packs Status page

License Pack 'station' Status		
Client ID	Address	Last Heartbeat
9b563dbae0ff4a78d00b473041da177	137.19.60.176	27-Jan-10 4:13:26 PM EST

The License Packs Status page shows the following information about each client that is currently leasing a license from the FLR:

- **Client ID**
This ID is specific to the application and changes with an application restart.
- **Address**
This is the IP address of the FLR Client host platform
- **Last Heartbeat**
This column displays the latest date and time that the FLR server received a heartbeat communication from the FLR client.

Reload the License Pack File (LPF)

Various situations may require you to reload the LPF in the FLR server application. This procedure describes how to use WorkbenchAX to reload an LPF in the FLR server application that is properly licensed, running, and where you have an established a WorkbenchAX to station connection.

- Step 1 In the WorkbenchAX nav tree, under the FLR Station node, expand the **Config** and **Services** nodes. The FloatingLicenseRepository component should display under the Services node.
- Step 2 In the nav tree, right-click on the FloatingLicenseRepository component and select **Actions > Reload** from the popup menu.
- The LPF is reloaded.
- The FloatingLicenseRepository component property sheet view displays.

Note: *When you reload the LPF all license pack counts are reset to zero at the FLR. FLR Client applications must reaquire licenses at their next heartbeat communication.*

Document change log

Updates (changes/additions) to this *Engineering Note: NiagaraAX Floating License Repository* document are listed below.

- Revised: June 12, 2010
Added a [Note](#) concerning the requirement for FLR server and client system clock synchronization.
Added the last line on first page copyright text.
- Publication: January 29, 2010
Initial publication.