iLabs InteractiveLabServer with LabVIEW Setup and Configuration

Last Modified: September 13, 2010

Included as part of the iLabs Interactive release is a sample Interactive LabServer with support for LabVIEW applications. The LabServer manages all authentication and communication with other iLabs services. Dynamic ASPX pages may be generated by the iLabs LabServer with support for embedding the LabVIEW remote panel in the created page, or a static HTML page may be hosted by either the .Net or LabVIEW Web Server.

The current release, iLab 3.0.4, includes support for the following LabVIEW releases, 2009, 8.6 and 8.2. By default LabVIEW 2009 is specified. This may be changed in the web.config file or on the LabExperiments page. Each labserver can only support one version of LabVIEW due to the limitations of the DCOM configuration required to connect to the LabVIEW VIServer.

To run this version you will need to have a supported version of the LabVIEW runtime installed with DataSocket server and at least one RemotePanel license. If you want to use a CGI VI for generating content you will need to install the Internet Toolkit (Part of the Enterprise toolkit in version 8.xx). Note behavior has changed since version 8.2 and the CGI interface does not seem to work when the LabVIEW Web Server and GWeb Server share the same port.

Two ports for HTTP are required, by default port 80 is for the .Net Web Services and port 81 is for the LabVIEW Web Server.

Currently the iLab LabServer must be installed on the machine with the LabVIEW installation used to run the experiment.

The .Net interface to LabVIEW is provided by .NET assemblies which wrap the LabVIEW activeX VIServer (see iLabs/LabView/LabViewInterface/), for each supported LabVIEW release an assembly is generated from these files. In addition a release specific collection of iLab specific VI’s are located in iLabs/LabVIEW/release/…/user.lib/iLabs. The release specific iLabs directory must be copied to the local Programfiles/National Instruments/LabVIEW release/user.lib directory.

A pre-built distribution of the InteractiveLabServer may be found at:
	iLabs\Builds\InteractiveLabServer

Source files may be found at:
    iLabs/Services/InteractiveLabServer – The interactive iLab Server
    iLabs/LabView/LabViewInterface82 – interface assembly

A VisualStudio 2005 Solution file for the entire ilabs Architecture with LabVIEW support for all thre releases is located at: iLabs/Projects/iLabProject/iLabProject_wLabVIEW.sln

iLab VirtualInstrument and support files are located in:
    iLabs/LabView/release/iLabsLvApp

Default URL is http://yourComputer/InteractiveLabServer

Interactive Lab Server with LabVIEW

Install LabVIEW

Install LabVIEW, DataSocket Server, RemotePanel Manager and any device drivers for your system. Once installed you must grant read and execute permissions on labview.exe to ASPNET or Network Authority

Configure LabVIEW

Enable tcp/ip viServer on the default port and allow access from all machines. Note only one installation of LabVIEW per machine may be on a specific port. The default port is 3363 and for this release it must be used, only one version of LabVIEW may use that port.

Configure the LabVIEW WebServer to run on port 81.

LabVIEW 8.2 should be the only release that uses the CGI Vis located in .../iLabs/LabVIEW/8.2/LViLabsApp/www/cgi-bin. If you will be using the CGI VI's install the LabVIEW Internet Toolkit and configure the LabVIEW WebServer to run on port 81 in shared port mode.

Configure LabVIEW DCOM

There are two sets of permission lists that must be edited to give the Interactive Lab Server the ability to launch our LabVIEW VI. Both are accessible through the Windows Component Services console.

Enabling access to launch DCOM objects:

This is required to ensure that DCOM control is enabled and that the account used by the Lab Server's ASP.NET application account can launch DCOM objects in general. NOTE: this does not give the ASP.NET application account the ability to launch any DCOM object. We still have to enable permissions on specific DCOM objects.

- Open: Control Panel>Administrative Tools>**Component Services**
- In the left-hand panel of the Component Services console, click on "Computers" and then select "My Computer"
- With "My Computer" selected, click on the "Configure My Computer" toolbar icon (). This should launch a configuration dialog.
- The "Default Properties" tab of the configuration dialog, ensure that the "Enable Distributed COM on this computer" item is enabled. By default, the "Default Authentication Level" should be set to "Connect" and the Default Impersonation Level" to "Identify". Click on the Apply button.

- In the “COM Security” tab of the configuration dialog, we must add the ASP.NET application account (either ASPNET or NT AUTHORITY\NETWORK SERVICE, depending on the Lab Server operating system) to the default launch and activation list. In the “Launch and Activation Permissions” portion of the dialog, click on the “Edit Default…” button. Add the ASP.NET application account to this list and give it full launch and activation abilities. Click OK to close that window. Do the same thing for the “Edit Limits…” lists in both the “Launch and Activation” and “Access Permissions” portions of the computer configuration dialog. When complete, click on the “Apply” button.
- Click “Ok” to close the configuration dialog

Enabling access to launch the LabVIEW Application DCOM Object:

This procedure will permit the ASP.NET application account to remotely launch LabVIEW through it’s associated DCOM Object.

- If it is not already open, open: Control Panel>Administrative Tools>Component Services
- In the left-hand panel, select Computers>My Computer>**DCOM Config**
- Right ClicK on LabVIEW Application, select properties
- Confirm that the path points to the correct LabVIEW Installation on my machine the path is: C:\Program Files\National Instruments\LabVIEW 2009\LabVIEW.exe /Automation
- **On Location:** Check; Run where the data is & Run application on This Computer
- **On Security:** for each section select Customize and edit the permissions, add the aspnet (XP) or network authority (Server 2003) user, grant this user all permissions.
- **Endpoints:** should be set to default system protocols, no change needed.
- **Identity:** Set to the interactive user.
- Click on “Apply” and then “Ok”
- Close all applications (be sure to save any un-saved work) and restart the computer.

Install iLab VirtualInstruments

Copy the entire directory from iLabs/LabView/release/iLabsLvApp/user.lib/iLabs to your LabVIEW installation’s user.lib directory. You may need to MassCompile this directory after it is installed to resolve any sub-VI relative paths.

To install the ILabs CGI VI:

Note: Not required or suggested for for any release except 8.2(at least for now).

Install the Internet Toolkit version 6.0, patch with update ‘LabVIEW InternetToolkit 6.0f2.zip’.

Check that the LabVIEW installation directory www/cgi-bin exists, if not create it.

Copy iLabs/LabView/8.2/iLabsLvApp/www/cgi-bin/ILAB_FrameContentCGI.vi to the LabVIEW installation cgi-bin directory.

Example GWebServer configuration files are located in the iLabs/LabView/8.2/iLabsLvApp/internet directory local versions of these files may already be in your LabVIEW home directory.

Convert Your Application:

There is an example application VI in iLabs/LabView/release/iLabsLvApp/Vis/Tank Simulation_DS.vi that shows how to modify a pre-built VI to use datasockets and the ESS. The VI is based on the example:

LabVIEW 8.6/examples/apps/tankmntr.llb/Tank Simulation.vi

.Net InteractiveLabServer Web Site

Use either the pre-built InteractiveLabServer or if you will be modifying your LabServer the source distribution

Create a virtual directory in IIS.

- Right-Click on **My Computer** (renamed to your computer name)
- Click **Manage**
- Expand **Services and Applications**
- Expand **Internet Information Services**
- Expand **Web Sites**
- Expand **Default Web Site**
- Right-click on **Default Web Site**
- Select New->Virtual Directory
- Click **Next** in the Virtual Directory Creation Wizard
- In the Alias box, type **InteractiveLabServer**

- Click **Next**
- Enter **D:\MIT_iCampus\iLabs\Services \InteractiveLabServer** in the Directory box (or click the Browse button, and navigate to the location where your Service Broker code is)
- Click **Next**
- Access Permissions: make sure you check the Run Scripts, Execute, Read and Browse permission boxes
- Click **Next**
- Click **Finish**
- You should see the **InteractiveLabServer** site listed under the Default Web Site
- Right click on the **InteractiveLabServer** website and select **Properties**
- Select the **ASP.NET** tab
- Use the pull-down list for the **ASP.NET** version and select 2.0.xxxxxx
- Select OK and close the properties window
- Close the Computer Management Window

Database Scripts

- Start-> **Enterprise Manager**
- Console Root->Microsoft SQL Servers->SQL Server Group->(local)->**Databases**
- Create a new Database, for example **iLab_ILS**
- Add a new user (yourMachine\ASPNET for XP or NETWORKAuthority for Server 2003), give this user owner permissions.
- Expand Databases
- Select the **ilab_ILS** database (or what you named it) in Enterprise Manager.
- On the menu bar, Select Tools -> **SQL Query Analyzer**
- In Query Analyzer, Click File->Open…
- Browse to database scripts directory typically (**D:\MIT iCampus\iLabs\ Database\DB_Scripts\ProcessAgent)**

Every service must have its own Process Agent information. These tables should exist before the service specific tables are added to each database.

- In the ProcessAgent directory select and open the **ProcessAgentTables.sql**
- Make sure the correct database **ilab_ILS** is the selected database in the dropdown at the top, then click the green triangle “play” button to run the script
- You should see a message about “The command(s) completed Successsfully.”
- Follow the same procedure to open and run the **ProcessAgentProcedures.sql** script.
- You should see a message about “The command(s) completed Successsfully.”
- Open and run the **SetDefaultsPA.sql** script. In Query Analyzer, Click File->Open…
- Browse to database scripts directory typically (**D:\MIT iCampus\iLabs\ Database\DB_Scripts\LabServer)**

The LabServer tables and procedures provide support for selecting, executing and presenting Lab experiments. They are not specific to LabVIEW.

- In the LabView directory select and open the **LabServer_Tables.sql**
- Make sure the correct database **ilab_ILS** is the selected database in the dropdown at the top, then click the green triangle “play” button to run the script
- You should see a message about “The command(s) completed Successsfully.”
- Follow the same procedure to open and run the **LabServer_Procedures.sql** script.
- You should see a message about “The command(s) completed Successsfully.”
- Later in the configuration process you will need to manually enter information into the LabServer tables.

Restart the IIS web server.

- Start -> Run
- Type iisreset

Web.Config

Web.config is a text file containing configuration information in XML format, residing in the root of the InteractiveLabServer web site (typically in the D:\MIT iCampus\iLabs\Services\InteractiveLabServer directory). Web.config.template ships with this release. If a web.config already exists, please delete it. Copy web.config.template to web.config and edit it using notepad. Note the database name must match the database that was built for this service, and may not be changed once the database is built.

Data from the web.config file will be used to populate the selfRegistration if the registration information has not been saved. You may modify all except the DefaultPassKey and Database Name in the SelfRegistration process.

sqlConnection	Database name
DefaultPassKey	Initial Passkey for credential exchange
serviceGuid	This is the ILS's Guid
serviceURL	Lab's service URL (InteractiveLabServer.asmx)
serviceName	Interactive Lab Server's unique name
codebaseURL	ILS root URL

Self Registration

In order for Process agents to exchange credentials, their information needs to be entered into the Process Agents table. There is a selfRegistration web page included with each process agent. You should not self register a process agent until after you have created and edited the web.config file for the process agent and created the database.

Use the information from the worksheet to fill in the correct data. Use fully qualified URLs

- Open Internet Explorer
- Browse to http://localhost/InteractiveLabServer/selfRegistration.aspx
- Modify any information
- You must have a unique GUID for the service, the UID generated is based on the Microsoft format but any globally unique string will do. I usually add a few human readable characters at the beginning of the string for ease in identifying the GUIDs in the database. GUIDs are limited to

50 characters. You may change the QUID up until the time that a ServiceBroker or other service is Registered.

Test the Interactive Lab Server:

- Open Internet Explorer
- Browse to http://localhost/InteractiveLabServer/InteractiveLabServer.asmx

Interactive Lab Server Lab Experiment Configuration

The example LabVIEW enabled InteractiveLabServer uses the database to configure individual experiments.

The interactiveLabServer database in addition to the ProcessAgent tables has the following tables:

- LabApp –- information about a specific experiment client, the only required data entry, see details below
- Task -- Used by system to track active tasks, no data entry required

The following tables are not used if you specify an application Key as part of the Client Loader Script (i.e. redirect URL) on the ServiceBroker. The function these tables originally supported are now managed by the Lab Side Scheduling (LSS) service.

- localGroup –- used to define local groups
- sbGroup –- uses a domain ServiceBroker and the SB’s group name to map to a local group.
- permissions –- assign an application ID to a local group. Currently one application per group, but could be a many to many relationship (would need to change LVPortal logic), roles could be used to trigger start-up and cool-down tasks for a group, currently I just use ‘run’ as the role but it is not checked.

There also are three experiment related tables, currently not used but may be used for temporary local storage.

LabApp Table:

The LabApp table contains any information required to run and present your Lab application. This table is managedusing the labExsperiments page. I have tried to not have any field dependant on

LabVIEW but to provide a structure that may support other application execution models. Information from this table is selected based on either the group permission or the specified application Key. All processing of the information is triggered by a ServiceBroker redirect to http://machineName/InteractiveLabServer/LVPortal.aspx (Note: this is specified on the ServiceBroker and may be changed in the 'Loader Script'. Please look at the codebehind and the LabViewTaskFactory class as an example of how to process the redirect. You may write your own portal page. Once an experiment is created a task is created to manage the experiment and the task is added to the taskList.

To access the LabApp Table Use the iLabServiceBroker, to access the Administer page of the LabServer, 'Lab Experiments' is the configuration page for experiments.

- LabApp_ID – primary Key internal database reference, do not use externally. Not displayed on page
- Title – Application Title, normally the Application Name displayed on the ServiceBroker.
- Version – This is independent from the LabVIEW version and may be used to denote changes in the application or different access levels to a specific client Required.
- Revision – internal information not used by the ServiceBroker. It may be used to specify a LabVIEW revision.
- Application Key - Key used as application specifier in redirect 'app=key', a LabServer unique key for the application.
- Path -- Absolute path to the directory or llb which contains the target application or VI. Do not include final path separator, use platform specific separators. Normally c:/Program Files/ National Instruments/LabVIEW revision/VIs
- Application – the name of the executable or FrontPanel VI, include extension.
- Web PageURL - the fully qualified URL of the page that will be presenting the VI or application, must not be null. See http:// yourMachine/InteractiveLabServer/RunExperiment.aspx. If you use dynamic page generation the following application URL is responsible for populating the region of the page displaying the LabVIEW FrontPanel.
- Application URL – Only needed if using Dynamic page generation. The default is http://yourMachine:81   See LVRemotePanel.ascx. Currently this is either the URL of the LabVIEW WebServer (for RunExperiment), or the URL of the CGI VI (for RunLab.aspx).
- Width – Application Frame width
- Height – Application Frame height

- DataSource – a comma delimited list of datasocket URLS, for each URL a datasocket writer and connection to the ESS will be created. An optional ‘recordType’ specification may be added to each datasocketURL by appending ‘=recordType’.

- Server – if LabVIEW is running on a remote machine specify hostname, else NULL. **Remote LabView server does not work in the current version**
- Port – if the LabVIEW viServer tcp/ip port is not the default set this to the port used. **Not fully tested in current version.**

- Description –- application description - Not used by LabServer***
- Contact –- an email or other contact info for the application ***
- Comment –- any comment currently not displayed ***
- ExtraInfo – optional XML encoded to be processed by your service (portal) page.

Running a LabVIEW InteractiveLabServer Experiment

The demonstration implementation of the LabVIEW Interactive LabServer uses the following process to present and manage experiments.

LabVIEW experiments are FrontPanel VI’s. They must be able to run from their specified location (path) and should not need to be saved after running. A change in LabVIEW 8.2 forces the requirement that they not be part of a project library.

If data is to be saved on an ESS the data must be written to a DataSocket and the DataSocketServer must be running on the labServer.

Enter data about the Lab Application in the LabApp table via Lab Experiments.aspx.

Register the experiment client on the service broker:

- Guid must match the values stored on the ServiceBroker, and USS & LSS for scheduling.
- The app query parameter in the loader script on the ServiceBroker must match appKey.
- Title and version do not need to match the values stored on the ServiceBroker, USS & LSS for scheduling.
- HTTP Redirect Client

http://MachineName/InteractiveLabServer/LVPortal.aspx?app=appKey

This is the default portal to all lab applications on the ILS, appKey is the ILS database AppKey for the specific lab application. If app is not specified the ExperimentExecution ticket will be parsed to find the ServiceBroker and SBGroup, which will be mapped to a localGroup and the application that group has permissions to run will be launched, replaced by LSS functionality. You may write your own portal page.

Launching the Client

From the ServiceBroker:

Once a user has selected the client and has met all required permissions to run the application, the service broker redirects to LVPortal.aspx, or another page specified via the client's 'Loader Script'.

On the LabServer:

Open LVPortal.aspx

- Get Tickets
- Validate authentication(s)
- If app is specified fetch the LabAppInfo for the application.
- If the app is not specified get the SBGroup & SBguid from the ExperimentExecution Ticket and query the database for the LabAppInfo for that group, the localGroup, sbGroup and permissions tables will need to have data.
- Pass the LabAppInfo,experiment coupon, and Execution ticket to the LabViewTaskFactory's createLabTask
    - createLabTask – parses the LabAppInfo and ExecutionTicket
    - connects to the VIServer
    - insures that the requested vi is loaded and ready to run
    - inserts a new Task into the database
    - if an ESS and datasocket(s) are specified
    - Create LabViewExp object
    - create readers for each DataSocket and add to the LabViewExp
    - add LabViewExp object to Global task hashTable, this object must be kept in memory for the readers to function.
    - taskPayload is constructed, added to database and task object
    - Add Task object to TaskProcessor
    - Return task
- Construct & store session information for the presentation page
- Redirect to the presentation page (RunExperiment.aspx or RunLab.aspx)

Parse Session information

RunExperiment.aspx -- Set LVPanel presentation data. LVPanel is a user defined (iLab) WebControl that generates an <object> block on the presentation page with parameters for the LabView RemotePanelControl.

RunLab.aspx -- Set LVFrame presentation data. LVFrame is a user defined (iLab) WebControl that generates an in-line frame from the output of ILAB_FrameContentCGI running on the LabVIEW web server.

Initialize JavaScript time remaining message

User runs application

Task Processor Thread

Every 10 seconds (default):

each task is checked for expiration, if expired it is moved to removeTask list

non-expired tasks have Heartbeat method called

each task in the removeTask list has its Expire method called and is then removed from list.

Task.Expire()

- Parses task payload
- Stops LabVIEW application (VI)
- Disconnects User from VI
- Checks for LabViewExp object if found
    - closes all DataSocket readers
- calls CancelTicket (ExecuteExperiment) on the SB
- Updates the database's task information