

Licas Getting Started Guide

Version 1.23

[A guide on how to install and quickly run an
application on the licas framework]

Kieran Greer, Email:

kgreer@distributedcomputingsystems.co.uk.

<http://distributedcomputingsystems.co.uk/licas.aspx>

Table of Contents

1	Introduction to the Licas System	3
2	Installing the Software	4
3	All-in-One GUI and Server	4
3.1	Start the GUI and Server	4
3.2	Load in a Message Service	6
4	Using a Service	8
4.1	Open a Service GUI Interface	8
4.2	Using the Service GUIs	9
4.3	Other Services	10
5	Running a Console Server	10
6	Running the Problem Solver	11
7	User Guides	11

1 Introduction to the Licas System

The licas system is an open source framework for building service-based networks, similar to what you would do on an Application Server platform. The framework comes with a server for running the services on, mechanisms for adding services to the server, mechanisms for linking services with each other, and mechanisms for allowing the services to communicate with each other. Functionality is provided to allow for XML-RPC and RESTful communication, and dynamic linking between services. The framework is very lightweight and the architecture and AI capabilities add something new that is not available in other similar systems. The open source server is mobile compatible and there is a mobile app available for Android systems. The All-in-One GUI is available from the main web site and can be used either as the platform for running your services, or for scientific testing. The server is also peer-to-peer, with the GUI both hosting services and acting as a client. The GUI is also designed to be used in one of two different ways, but because of resource constraints, the applications have been packaged together. It can firstly be used for loading and running services and all of the required administration. It can then also be used to run more complex scientific applications. The scientific panels are hidden initially, but you can open them, configure and run tests and display graph results, for example.

The GUI can perform a range of functions, including service admin, using services through their own interactive interfaces, or running tests. So it includes all of the modules and packages that are available in the source code. If you are not interested in the AI testing, you can ignore the scientific panels that are not opened as default. The application also allows you to load your own service modules, view and interact with them. The GUI can also link with other licas servers and view those configurations as well. There is a level of password protection however, to restrict full access to a remote server. Communication with remote servers comes in the form of a metadata (XML) description that is then displayed as a view in the graphic. These XML-based descriptions mean that it should be possible to create a SOA on some server and view it remotely.

A network is a group of distributed servers and the term SOA (Service-Oriented Architecture) is being used to describe the setup at a single server. That is - a base server running an ESB (Enterprise Service Bus) that stores and runs any number of services. The terms are used a bit loosely here, but fall into that general architecture. Servers can therefore link to create a network, or services can link on a server or across servers, to provide the SOA environment. A number of default services are provided, with some useful functionality, see the 'licasService' guide. A modular setup and configuration script then allows you to add other modules. The server source code can be downloaded from sourceforge and includes the main packages. There is then a setup installer for windows

that will install the All-in-One GUI and its related files, which includes all of the platform packages in the one system.

2 Installing the Software

The free All-in-One GUI installs using the `setup.exe` only, downloadable from <https://distributedcomputingsystems.co.uk/licas.html>. The installer installs the main program into a `DCS/Licas` folder in `Program Files`. It will add the program to your `Startup` menu under the `DCS` name, and as a shortcut icon on your desktop. A user folder called `licasData` is also created and installed in your root user directory, for example, `'C:/Users/Your Name/licasData'`. This folder is for data and configuration files, but also external program jar files, such as services and any external modules. The `web` folder is also added to `licasData`. It has limited use, to allow access to files that can be searched or retrieved using online protocols. Finally, the program actually runs from the `cmd` folder in `licasData`, so please take some care before changing it. You can still use the `licasData` folder for general use, to store any of your own files, where the 'hidden' folders should be left alone. After starting the GUI, you need to load in the admin configuration and start the server before loading any services.

3 All-in-One GUI and Server

If you don't intend to program and want to use the system simply for running applications, you can use the All-in-One GUI. After opening the GUI, an 'Admin / Load GUI Config' menu option allows you to read in the config information. Information about default services is also stored there. A service is represented graphically by a node in the Server panel of the GUI. If you right-click on the node, you can then send some commands to the service through this view. Communication with the service itself can be carried out through XML-based descriptions that use Java Reflection and method invocation. The following sections describe this further.

3.1 Start the GUI and Server

The All-in-One GUI can be started either by double clicking on the desktop icon, or through the `Startup` menu option `DCS/Licas/Licas`. The set of default services might be quite useful and so you might setup a service configuration and then save it as a stored SOA. When you go to use the GUI, you might then want to simply run the services again and not

bother with the whole application. To help with this, when the application starts it displays a small menu, as shown in Figure 1.

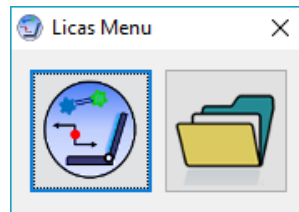


Figure 1. Startup Menu.

If this is your first use of the system, then you should click the LHS button, which opens the main GUI, as shown in Figure 2. If you have already used the application and setup a service configuration, you can click the RHS button to browse to your saved config and load it in directly. The main GUI is still loaded in the background, but this second setup displays a smaller Service Console and all of the services. Something like Figure 5 would be loaded.

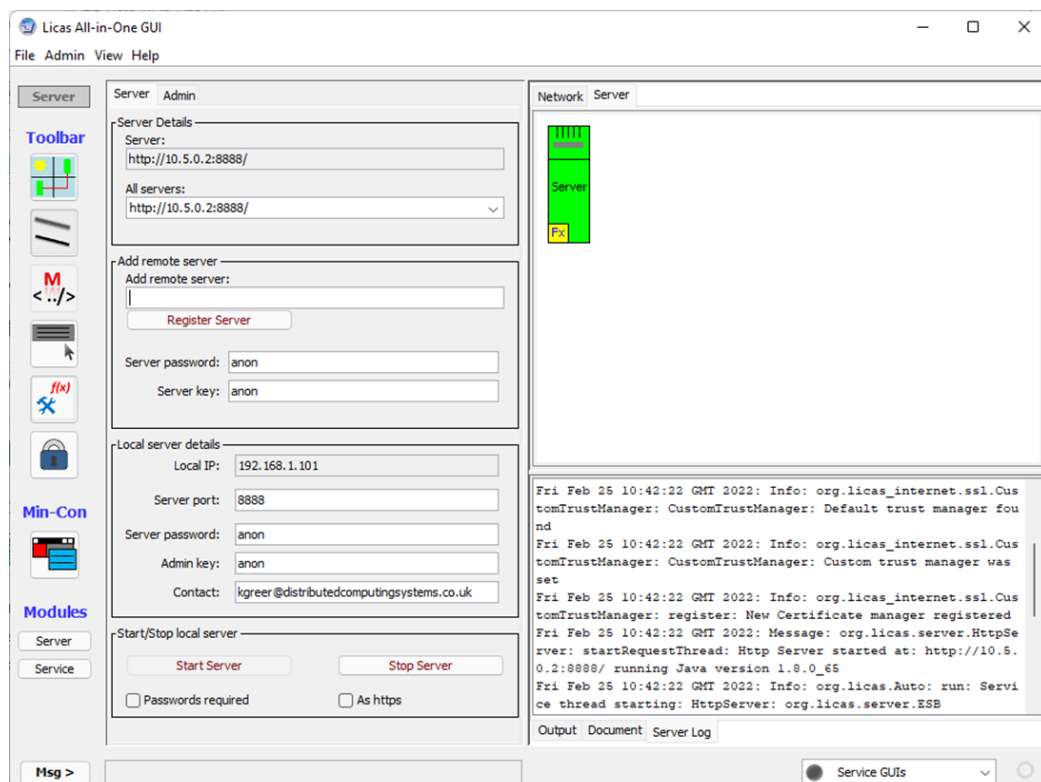


Figure 2. Server panel and tab with server started.

Figure 2 gives an example of the server running. A `Server` panel can be accessed by clicking on the `Server` module button. It should also be the first panel that is displayed. After loading in the GUI config, if you click on the `Start Server` button, this should start a server. There is also a `Server Log` tab in the bottom right quadrant that gives output of what the server is doing. There is also a `Refresh View` toolbar button (top left) that will update the network view when it is clicked. The `Server Details` section of the panel should display the `ip address` that the server is running on. When you load in a configuration or start the server, some messages are displayed, so click OK to those. The first configuration also loads the `Service` and `Jar Factories` onto the server. These can be seen with additional `Fx` or `J` squares at the bottom. The `Jar Factory` can process requests for jar files from remote servers, but these can be disabled. In the bottom right corner, there is a combo box that lists any of the service GUI interfaces that have been loaded. If you open any service interface and minimise it, you can then use this list to quickly find and re-open the service interface again. If you right-click on any graphic item, it should open a popup menu, with options for interacting with the server component.

3.2 Loading a Service onto the Server

The following steps will load a service onto the server and are explained further in the following sections:

1. After starting the All-in-One GUI, select the `Open Main GUI` button from the first menu.
2. From the Admin menu, select the `Load GUI Config` option. This also loads in your default services.
3. Change the server port and passwords if desired and then click `Start Server` to start the local server.
4. From the `Modules` column, click the `Service` button.
5. Select the top RHS `Server` graphic and click the `licas` server icon. This should add the name `HttpServer` to the top selected service box.
6. From the `All service types` combo box, select the service type, `Message`, for example.
7. Add a `Service ID` as the user name for the service.
8. Change the service passwords if desired.
9. Click `Load Service` to load it onto the `licas` server.
10. Clicking on the top button in the `Toolbar` column will refresh the graphic.
11. Right-click on the service icon to open its GUI.

3.3 Load in a Message Service

After starting the server, you can run a service. Loading a service can be carried out with a minimal amount of effort. If the admin config file has been read (Admin / Load GUI config), a list of services will be displayed in the 'Service' panel. If you click the `Service` modules button, a Service panel should be displayed. This provides options to allow you to add any of the registered services onto the server, as is shown in Figure 3.

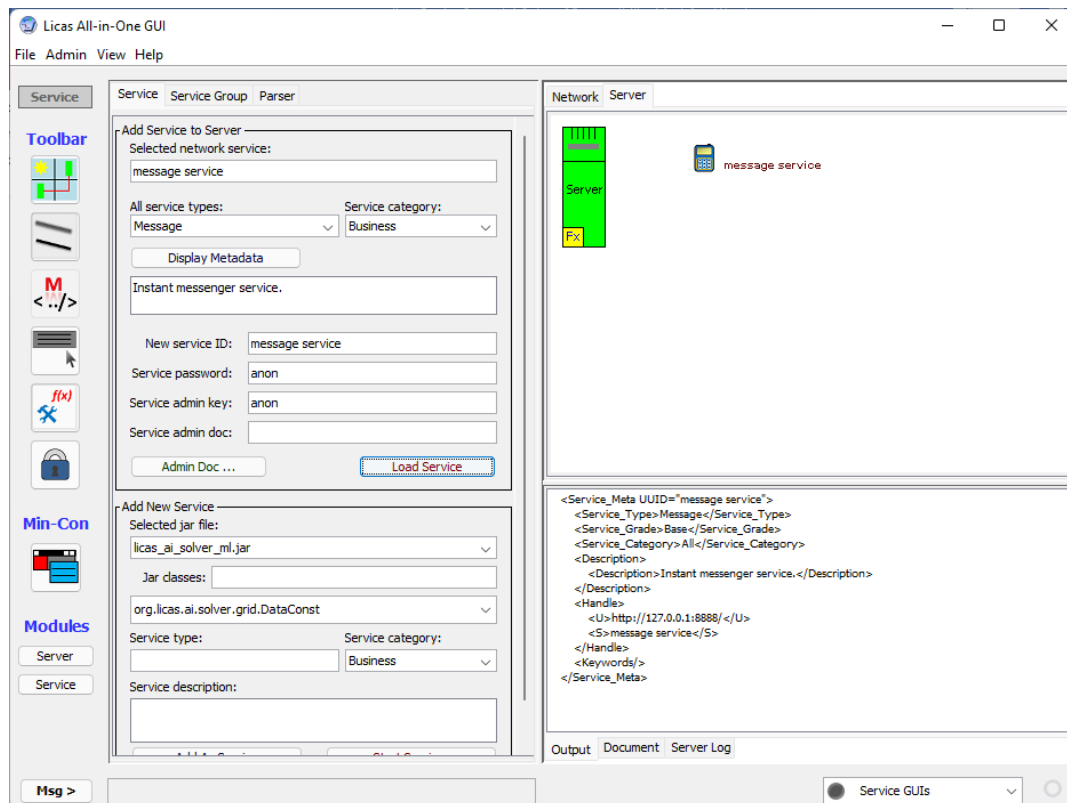


Figure 3. Service panel with default message service loaded.

The services are listed under categories and types. Categories relate to general topics, like business or personal, etc. The type is then the service function, for example, `File` or `Message`. You can select the `Message` service type from this combo box. Any service needs to be added to another service, which includes the server. To indicate the server, click on the server component in the graphic view. That is the green computer graphic with the word 'Server' written on it. This should add the word 'HttpServer' to the `Selected network service` text field. You then need to assign to your own service a unique name or id. In the `New service ID` text box, type in the name to assign to the service. If you then click the `Load Service` button, you should get a message confirming that the service has been added to the server. The 'refresh network view' `Toolbar` button (the top left button) will refresh the view. Figure 3 is an example of a message service being displayed in the server view. There are also options to initialise the service with passwords

and an admin script. For that you select the constructor and add the parameter values. As this is slightly more complicated, see the 'licasGui' document for more details. If you place the mouse over the graphic of the service and right-click it, you should open up a pop-up menu with different options. A `ServiceFactory` can help to configure this. See the main 'licasGui' document for more details.

4 Using a Service

A default instant messenger service is provided as part of the installation package and is relatively easy to use, where the main problem is adding the correct http address and passwords, for any remote service that you wish to communicate with. As illustrated in Figure 3., you can load services onto a server through the Service panel. The `Business` category includes a `Message` service. To use all of the default settings, you only need to enter a service name and click the `Load Service` button to load one. A confirmation message should be displayed.

4.1 Open a Service GUI Interface

If you right-click on the service in the server graphic, a pop-up menu should show a GUI option. If you click that option, then if there is a GUI associated with the service type, it will open. If there is no GUI associated with the service, you will receive a message instead. For the message service, the default interface should open, as shown in Figure 4. The message service interface will initially open with an empty address book, where you can add addresses by clicking on the `Add Address` button. The addresses are saved as an XML file in the default 'licasData' folder. Note that the address details are for the remote person that you want to communicate with, not your own local server.

To use the system, each service needs to know the name, location and password of the service that it wants to call. The passwords include a password for the remote server and a password for the remote service itself. This is the basic security provided by the system, as otherwise any service is open to be invoked. However, if you do not enter or change the password settings, then they default to 'anon' and so the address book opens with these default values. The username that gets used is also the name of the service. This is 'hercule_local' in this case and for the message service to be called, it is 'duke_local'. Then in the address book, the server address is the IP address of the remote server, which requires an ip number and a port number. Note that if the connection is to a computer outside of the internal network, your firewall might need to be configured to allow connections through it. After the address details have been added, you can click the add

address button to save them permanently and also write them to the address book file. You can then click the `Close` button to close the address book form. The next section describes how to use this application to pass messages to another message service, running at some other location.

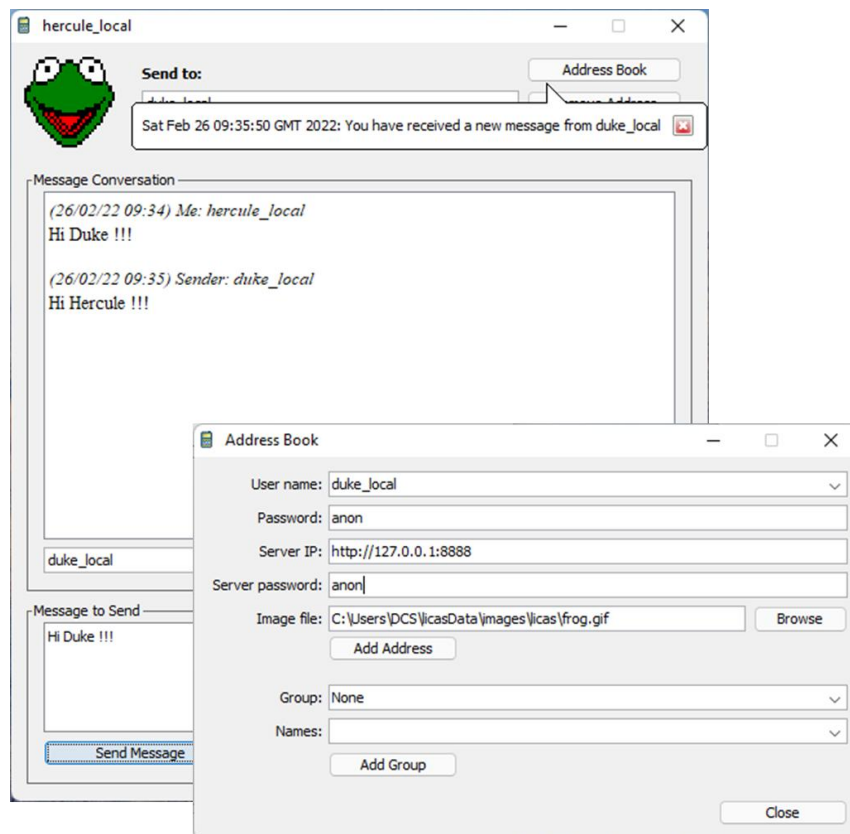


Figure 4. New message service window, also with address book form.

4.2 Using the Service GUIs

If you perform the actions of the last section on separate computers, you will setup the licas environment with message services that can communicate with each other. You can then select the correct address for a remote client and you will have an instant message service. Figure 5 shows a suite of services running, including the instant message service. The other services are a file sharing and an Information service. The main GUI has been minimised to the service console (Min-Con toolbar button), that allows you simply to open any of the service windows from the selection box.

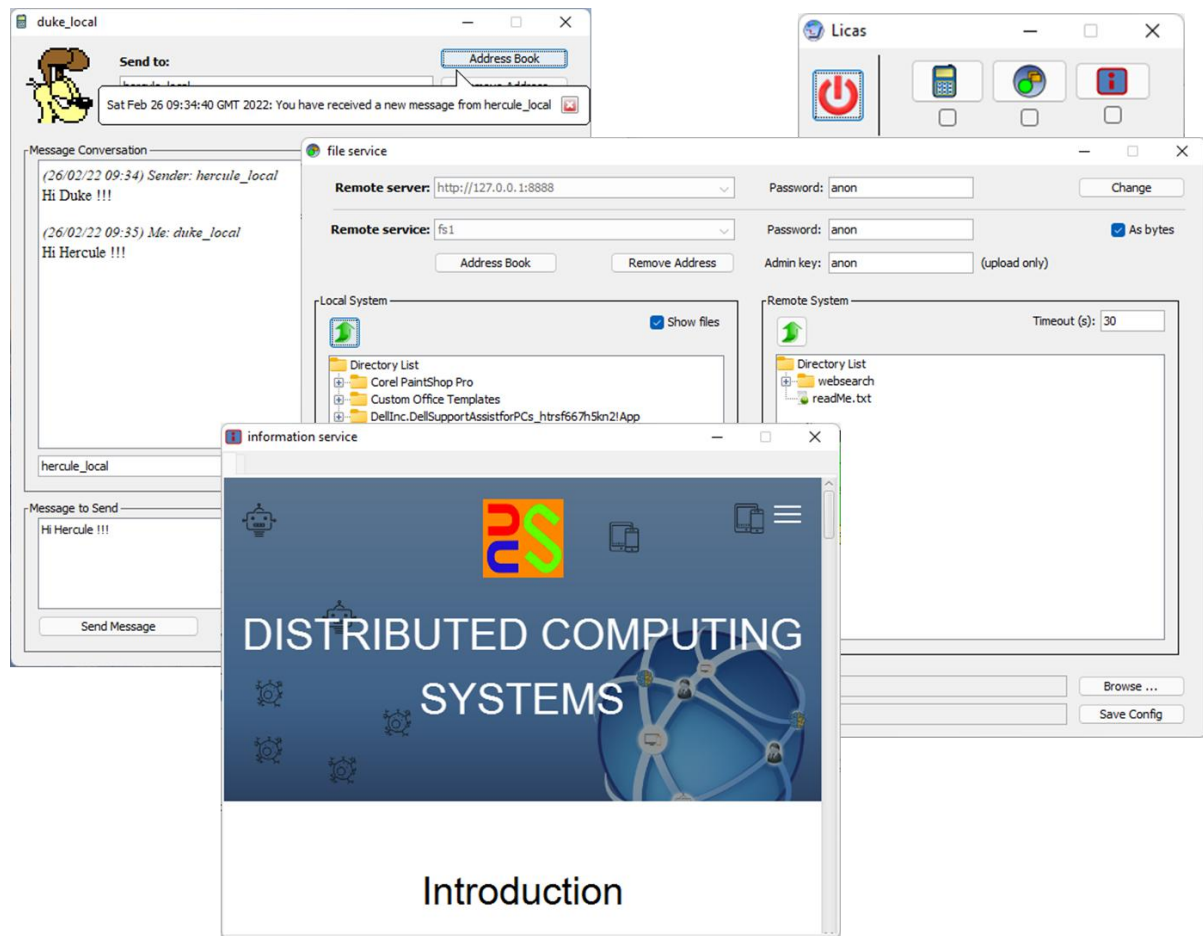


Figure 5. Server running 3 services, including the Instant message service.

4.3 Other Services

For the other services, see the 'licasService' guide.

5 Running a Console Server

The open-source package includes a stand-alone server that can be started from a command box. To run a server, simply run the batch or script file that is included in the server download folder. The server is now compiled under Java SE version 8, but it is also Android compatible. All you need to include is a class to start the server running. You can copy the `Run_HttpServer` class into your J2SE project and start it from there, for example. The batch file opens a command box when the server is running.

There is a log file, which is initialised with a configuration file that can be found in the configuration folder and is called 'logger.config'. You can change this to change the log file settings, to write to different streams, etc. The log file used is the jlog2 package that is very basic, but sufficient for this application.

6 Running the Problem Solver

A problem solver package has been included that is also open source and extends the core licas package. It is also integrated into the All-in-One GUI, where you can configure and run test scripts and even view some results in graphs. The 'licasProblemSolver' document describes in detail how to use the problem solver. The jar file is stored in the solver folder and the `runTest(Com/Server)` batch or script file will run a test as specified by the configuration. The GUI provides much more functionality however. It still tries to run a script, but the script can be configured through the GUI instead of manually and you can set all of the values through the GUI and display a view of the result. You can also start the client separately and then run the problem solver batch file. Note the relative paths for files. While the system is distributed, it can be resource-expensive and so only a limited number of services are recommended for a problem-solver.

7 User Guides

There are a number of documents provided with the licas system. The content of these is as follows:

- The `licasGettingStarted` guide (this document) describes the different installation options.
- The `licasArchitecture` document gives a general overview of the whole system structure. This describes how the service components are connected in a network and how to communicate with or link to them.
- The `licasGui` document describes the default GUI application that is provided with the system. It is not open source, but is provided for free, for testing or non-commercial use. The document describes the different panels in the GUI and how to use them. You do not need all of these to run a network or test the system, where the feature set is now quite rich.
- The `licasService` document describes how you can add your own services or behaviours to the system. There is also a detailed description of how to add an interface to a service, turning it into a Cloud or SOA application.

- The `licasAdminGuide` document describes the metadata structure for describing the services. This includes the internal metadata describing the service's functionality, or metadata for service level agreements and security. You only need to worry about the metadata if you intend to test the security features or if you are interested in testing semantic processing. You do not need any additional metadata to create a network and invoke the services on it.
- The `licasUserGuide` document describes in detail some of the programming methods for writing code using the system. You will need to read this before you can do any programming. It is relatively easy to use the system once you have grasped the general idea of how to add a service and then communicate with it.
- The `licasAuto` guide gives details on how to use the Autonomic Manager and configure scripts for it.
- The `licasProblemSolver` document describes how to use the problem solving framework that has been written on-top of the base `licas` packages.