

Licas All-in-One Gui

Version 6.3.1

[A guide to using the GUI application for running basic services on the licas system]

Kieran Greer,

Email: kgreer@distributedcomputingsystems.co.uk.

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

Table of Contents

1	Introduction	4
1.1	Application Overview	4
1.1.1	Interactive Graphical Display	5
1.1.2	Controls.....	6
1.1.3	Start-up Sequence.....	6
1.2	Services Console.....	7
1.2.1	Auto-Hide Services.....	7
1.3	Menu Options.....	7
1.3.1	File Menu	8
1.3.2	Admin Menu	8
1.3.3	View Menu	9
1.3.4	Help Menu	10
1.4	Main Form ToolBar.....	10
1.5	Popup Menus	11
1.6	Button colouring.....	11
2	Server Panel	12
2.1	Server Details Group Box	12
2.2	Add Remote Server Group Box	13
2.3	Local Server Details Group Box	14
2.4	Start / Stop Server Group Box.....	14
2.5	GUI and Service Configuration	15
2.5.1	GUI Admin Details.....	17
2.5.2	Service Factory and Jar Factory	17
2.5.3	Specify new Service or GUI Interface.....	18
2.5.4	Function Services Configuration	18
2.5.5	External Modules	19
2.5.6	Service Icons.....	21
2.5.7	Adding New Default Services.....	22
2.5.8	Copying Scripts.....	22
2.6	Service Admin Document.....	22
2.7	Security and Web Manager.....	22
2.8	Server Popup Menu.....	24
3	Admin Panel.....	26
3.1	Server Admin Group Box.....	26
3.2	Allow / Block Add Services	26
3.3	Server Graphic Group Box.....	27
3.4	Enable Scientific Panels.....	28
3.5	Save GUI Config.....	29
4	Service Panel.....	30

4.1	Add Service to Server Group Box	31
4.1.1	Service Constructor	31
4.2	Add New Service Group Box	31
4.3	Use the Service Factory to define a new Service	32
4.4	Popup Menu Options	33
5	Service Group Panel	35
6	Parser Panel	36
6.1	Parser Classes Group Box	37
6.2	Template Classes Group Box	37
6.3	Load Class Group Box	38
7	Method Panel	39
7.1	Selected Service Group Box	39
7.2	Method Details Group Box	40
7.2.1	Other Method Info Form	40
7.3	Execute Method Group Box	41

1 Introduction

This guide describes the free All-in-One GUI application, available for general use. The application includes all of the modules and packages that are available and runs as a p2p server with an interactive interface. The scientific panels are now described in a separate 'licasGuiScientific' document, but you need to read this document first. The GUI also allows you to load and run your own services, and interact with them. Details of any components that are running can be viewed in a graphical display. The admin panels can also register other servers and provide a view of those configurations as well.

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, where the services and the server can communicate with each other, query, or link to form associations. Servers can therefore link to create a network, or services can link on a server or across servers, to provide the SOA environment. This is therefore like a web application environment that it is private to your own computer, but with additional functionality to run more scientific processes. A number of default services are provided for everyday use, see the 'licasService' guide. You can however, also add your own services through a minimal amount of additional programming, where a modular setup and configuration script allows the details to be saved and retrieved. The GUI can therefore be used in one of two ways. The first is as a private SOA or Cloud environment for your own services and the other is as a test platform for performing distributed or AI-related tests. The main features of the Business GUI and general setup are described in the following sections.

1.1 Application Overview

The GUI consists of:

- A toolbar on the left-hand side.
- A number of tabbed panels for each GUI function that can be selected or opened using the toolbar.
- A graphical area for displaying and interacting with the network components.
- A number of text areas below that, for interactive feedback or logging information.

Figure 1 shows the application with the network panel visible. Note that both 'http' and 'https' protocols are possible, where selecting the `As https` check box will start the server using an https protocol.

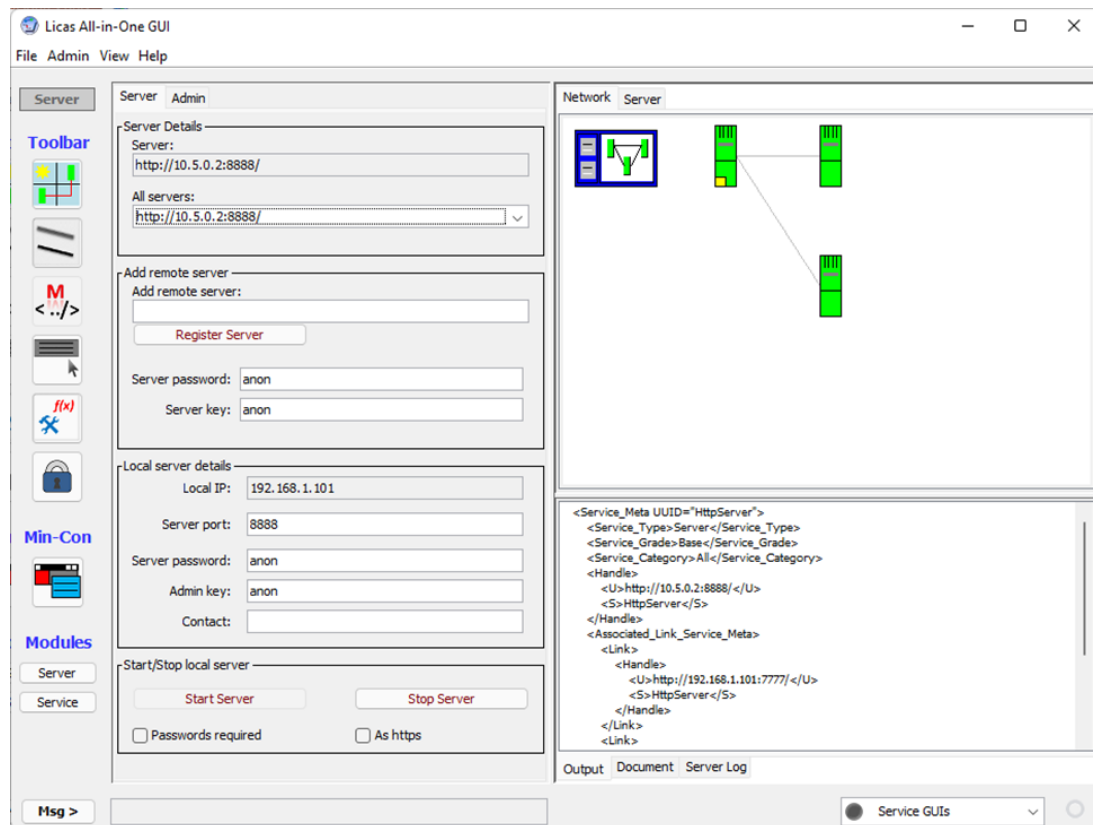


Figure 1. Licas GUI with network graphic showing.

1.1.1 Interactive Graphical Display

The upper RHS tabbed panes give graphical views of either the servers that are registered, or the services on a single server.

1.1.1.1 Network View

This is the view shown in Figure 1. Three servers have been registered with the GUI and are linked as shown. The yellow square indicates the local server that has a link with the local service factory. The popup menu then allows permanent links between the servers. These links translate to knowledge of what servers know about other servers. They can be created or removed dynamically and that will update an associated links setting on the server itself. There is a level of password protection, to restrict full access to a remote server, where communication with remote servers comes in the form of a metadata (XML) description that is then displayed in the view. It only displays a 'view' of the network, based on the XML descriptions and not full interaction. These XML-based descriptions however mean that it should be possible to create your own network of services built on the base licas classes and view them.

1.1.1.2 Server View

The server panel graphic is hidden behind the network panel but is shown in the other figures. It gives a view of a single server, displaying all of the services running on it. The services can be connected through different types of link and there is a more detailed list of popup options with that panel, for displaying metadata or manipulating the GUI components. See the 'licasAdminGuide' document for more information on the different types of link or metadata.

1.1.2 Controls

There is a toolbar on the left-hand side of the application that runs vertically. This has some `Toolbar` buttons for easy access to certain graphical operations. It also contains a list of `Module` buttons that display or hide function panels when they are clicked. Instead of all panels being displayed, they have been separated into different categories, to make it easier to use, where the menu options can open up more tabs. On the Server panel, there are also some server protection options. The first is a `Passwords required` check box. If this is not selected, then the server is essentially open and will always allow access to its base methods from an appropriate request. If it is password protected however, then to call the server, you need to know the server password. The second check box is `As https`, which formats the server URI as an HTTPS address, where the default is just HTTP. Another addition is a combo box on the left-hand side of the GUI. If you open up interfaces for any services, the service UUID should be added to this combo box. If you then minimise the service interface GUI, you can select its UUID from the combo box to maximise it again and bring it back into view. Finally, there is a text area with a `Next` button. The GUI may output some information relating to a current operation in the text area. You can then move to the next message using the `Next` button.

1.1.3 Start-up Sequence

To fully initialise the GUI, you would typically open it from the desktop and then:

1. The first thing is to load in a configuration from the `Admin` menu (see next section). Typically this would be the default configuration and when you first start the GUI, one should be created.
2. Before you can run any services, you need to start the server. The configuration can add the default settings, such as the server passwords or port number, for example. You then click the `Start Server` button to start a server.
3. The GUI is now fully initialised and service can be added and run.

1.2 Services Console

The main GUI is quite large and so there is an option to hide it and open a much smaller 'service console' that lets you access each service GUI through a simplified interface. The minimised console is shown in Figure 2 and consists of a list of running services and a button to shut-down the system. There is a toolbar button on the main GUI to open the console, under the `Min-Con` label, or you can select the minimum console at the start. When closing the minimum console, you then have the option to exit the system completely, or to revert back to the main GUI again. You would therefore configure and setup your services using the main GUI, but then maybe save that as an SOA, load it in again and use it with the minimum console.

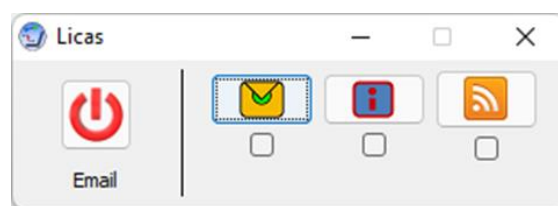


Figure 2. Minimised Services Console

1.2.1 Auto-Hide Services

When using the minimum console, you select one of the services from the apps list. This list shows a button and a check box for each available app GUI. If you click on the button, it will bring that app to the front. If you move to a button, it displays the app name and if there is a scroll bar and you scroll through the apps, the last visible app name is again displayed. If you click the check box and then the button, it will auto-hide the app GUI. This would mean that the GUI is hidden until you move the mouse over it. Then it is shown and you can interact with it. You can then unclick the check box to permanently show the app again. After any change, click the app button to set it.

1.3 Menu Options

The application also contains a menu bar with options to load either a document or a network into the system. A network can be loaded onto the server that it was saved from. Similar options exist for saving the existing document or network to a file. Documents can be saved or loaded as text (.txt) or XML (.xml) files, while the network config is saved as an XML (.xml) file. If saving a network, the saved file does not store all of the network

information, but it is useful and can be used to restore more basic network configurations automatically. The following sections describe the menu options that are available.

1.3.1 File Menu

This contains options for loading or saving files. The following options are available:

- **Load SOA:** this option allows you to read a set of files that describe a previously setup local server and services. These contain an XML-based description of a configuration of services, or SOA, running on the server and the parser tries to recreate or load the setup automatically onto the server again. **Note:** you can now select this option immediately after starting the GUI and if the default config is not loaded, or the server is not running, those operations will be automatically carried out first.
- **Save SOA:** this option allows you to convert the currently running SOA, or services configuration, into a set of XML-based files and save them locally, so that the configuration can be re-loaded at some other time.
- **Load Network:** this option allows you to read a file that describes a previous network setup. It will try to register remote servers that are not currently running. It will also try to add links from the local server to the remote one only.
- **Save Network:** this option allows you to save a current network setup to XML. The addresses and passwords of the servers that are currently registered with the GUI get saved. The servers' links with each other are also saved, but only the local server links are re-established when the file is read in again.
- **Load File:** this option allows you load any type of file. It will be read and then displayed in the `Document` tab text area.
- **Save File:** this option allows you to save the contents of the `Document` tab text area to a file.

1.3.2 Admin Menu

This contains options for configuring the GUI. The following options are available:

- **Load Gui Config:** this reads the saved GUI config file and tries to load in the values into the GUI. The default file has the name `default_config.xml` and should not be changed. It is saved to the 'licasData/config' folder, where you should save all of your config files for convenience. If you have saved service factory settings, for example, this option will try to load in the specified service factory, together with any related services or modules. The service factory form (see later) can then be used for manually changing the details again. The GUI config also stores the server admin passwords, or specific graphic settings, for example, which is why you might want to save different setups.
- **Choose Gui Config:** this first asks you to browse to the config file that you want to load. So it can be a different file to the default one and allows you to store different

configurations. For a new configuration, it would be best to load in the default one first, update it and then save the updated version as the new config.

- **Save Gui Config:** this saves the current server and admin panel settings to the default config file.
- **Save Gui Config As ...:** this allows you to save the current config settings to a new file, so that you can create different versions.
- **Open Service Factory:** this option opens the GUI form for loading in or changing a service factory object, or any other config details. If you add new modules, or change the factory settings, for example, it should be done immediately after opening the GUI and then the GUI should be closed again and re-opened. It should then load the new settings into to the other panels.
- **Show Passwords:** this option will list all currently stored passwords in the main text area. The format is to display the ip address and then the service ID, without the additional handle XML tags. It is still stored in the system as a full XML-based set of elements.
- **Clear All Passwords:** this is an override option if the network becomes corrupted and passwords are stored for services that do not exist. This can then be used to try to clear all passwords except those for the servers themselves. So any other service passwords on any server will all be removed.
- **Remove Passwords:** if a process becomes faulty, it may be necessary to manually remove passwords related to a server or service. This option opens a form that allows you to enter the server or service details, when any stored passwords related to that are removed.
- **Remove Passwords:** this option opens the same form, but allows you to change or update the passwords that are stored for a specified server or service. Note that you can use Show passwords first, to see what IDs to enter.
- **Glass Pane:** this option opens a glass pane over the whole GUI. The idea being that you can run the GUI but will not accidentally click one of the buttons. The Window form's `Close` option is still available however. You can then right-click the GUI and use the popup menu option to remove the glass pane.

1.3.3View Menu

This contains options for changing the GUI view. The following options are available:

- **Compact:** this option shrinks the GUI to a set smaller size, suitable for running just one or two services.
- **Full Screen:** this option enlarges the GUI to the full screen width, but the height remains the default size.
- **Scientific Panel:** this option adds the Scientific button to the module buttons options. You can then click on it to view the Scientific panels. This can also be de-selected.
- **Utility:** this option adds the Utility button to the module buttons options. You can then click on it to view the Utility panels. This can also be de-selected.

1.3.4 Help Menu

This contains options for displaying help or checking you applications version. The following options are available:

- **About:** this opens an about box with some general information.

1.4 Main Form ToolBar

A toolbar runs down the left-hand side of the application. This gives access to certain graphical features as described next:



Refresh View: The application is now set to refresh the view of the network every 60 seconds. You can however click this button to obtain an immediate view refresh. The network graphic will immediately be redrawn.



Show Links: If this toggle button is set, then if you move the mouse over a network component, its links to other components will be highlighted.



Show Metadata: If you click on a network component, its metadata can be retrieved and displayed in the output window. This now only happens if this toggle button is set. If it is not set, then you can click on a service (component) but no metadata will be displayed.



Show Tooltips: If this toggle button is set, then if you hover over a graphic component, info in the form of a tooltip will automatically be displayed. Things such as the component's ID, path or description will be shown.



Load Service Factory: The application can now attach or associate separate GUI interfaces with particular services. Before this can happen, a service factory needs to be loaded, to tell the application what interfaces belong to which services. This button allows the service factory object to be loaded.



Security and Web Manager: This opens a form allowing you to configure or select what folders are publically available as part of a web server. The default setting creates a 'web' folder in the installation directory and only allows HTTP requests to retrieve files directly from that location. You can then change this through the GUI form. The logger settings can also be changed.



Minimum Console: This opens the minimum service console form and hides the main GUI. From the minimum console, you are provided with a list of services that have active GUI interfaces and it allows you to easily select from them. There is then the option to toggle back to the main GUI again or shut the system down completely.

1.5 Popup Menus

The server panel in particular, now has a more complete set of commands from its popup menu. See section 4.4 for more details about this.

1.6 Button colouring

Because the GUI is quite complex, the buttons are now colour coded, to help with identifying the correct use of a button. The colour coding goes as follows:

- **Blue:** means an operation on the GUI (local) itself.
- **Red:** means an operation on any running network.
- **Green:** means to add a value (method parameter) to any description that is being constructed.

2 Server Panel

Figure 3 shows the GUI opened with the server panel visible. The server panel can be used to view a local or remote server, with an interactive display. The server graphic is displayed in the top right panel, while the bottom right panel is another tabbed pane that can display different types of text. One panel there displays an output trace of what GUI options have been selected, another displays textual documents that have been loaded, while the third text area provides a trace for a locally running server. There is also a message box on the bottom toolbar or status bar. This might occasionally output messages or information. You click the `Next` button to move to the next message. You can see that the nodes are staggered slightly, for the same level in a hierarchy. This helps to separate out the link lines when there are a larger number of nodes at one particular level. It is only a graphic feature and does not affect anything.

A local server can be started or remote ones registered.

- The simplest way to start a server running is to click the `Start Server` button in the `Start/Stop local server` group box. This will start a server running in the GUI application itself. The `Server` text area will then display the output that the server is generating.
- If the `Passwords required` box is checked, this will protect the server from external invocations and will always require the correct password. The GUI automatically stores its passwords and has administrator privileges. If the `Passwords required` box is not checked (default) then upon request, the server will always allow method invocations from a correctly formatted request. Any services would still have their own set of passwords for protection, as they might not be owned by the server.
- If the `As https` box is checked, then an HTTPS protocol is used instead of the default HTTP one. This is supposed to be a more secure protocol.
- The `Add remote server` group box then allows you to register a remote server. For this you need the URI (`http`) address that the server is running on and its password, and optional service admin key. These default to `anon`, but they can be changed to what is required.

2.1 Server Details Group Box

This is the top most group box and displays details on what server the GUI is currently connected to and displaying. The `All servers` combo box stores the addresses of all servers that are registered. You can then change the server connection by selecting a different address there. Any new server that is registered should also be added to this list

and when it is removed it should also be removed from this list. If you right-click the combo you can remove the currently selected server.

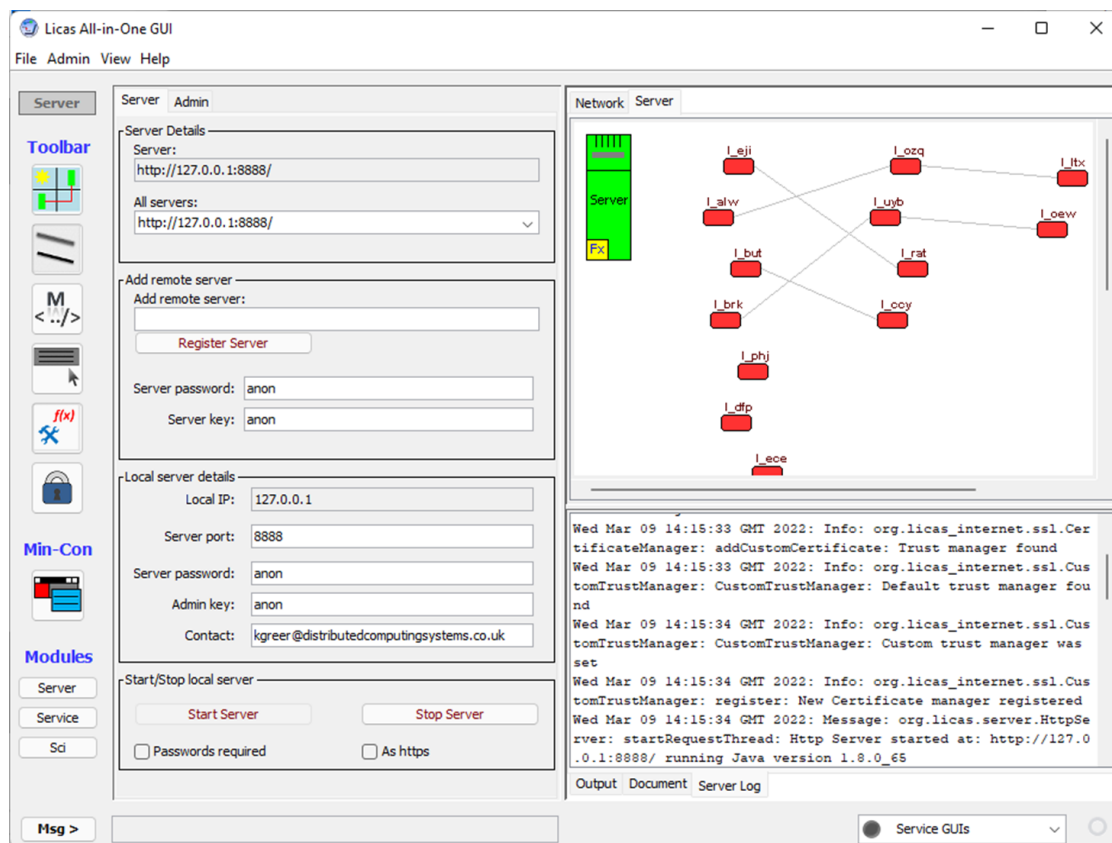


Figure 3: Server Panel

2.2 Add Remote Server Group Box

This allows you to register and connect to a remote server. To connect, you need to type the server URI into the Add remote server text box and also enter the required passwords. This would be, for example: `http(s)://ip_address:port_number`. For example: `http://127.0.0.1:8888` for the same computer. Two other boxes allow you to enter the password and service admin key values. These default to `anon`, but can be changed to any value. This means that the remote server can still be protected with passwords and only accessed by allowed users or the administrator. After a password is known however, it provides automatic access, unless an admin script has configured different access levels.

If it is possible to connect to the server, then the server URI will be displayed in the `Server` text box and if a server exists, the metadata will be retrieved and displayed. A local SOA can be seen in the top right panel, where the server is drawn in green and some linked services in red. The top-level services are drawn in red, while child services are drawn in grey.

2.3 Local Server Details Group Box

This allows the user to start a server running locally in the GUI itself. This means that you can run local tests, or act as both a server and a client (p2p). The server details displayed here get saved with the whole GUI config. They have been recently expanded to include a server administrator contact address. This might be useful to a remote user and can be returned from the server itself, as part of some interaction. While this is saved with the server port and passwords, the ip address field is not saved but gets set by the computer when the GUI loads. You can however change it to some other value if aliases are used and when you start the local server, the alias address should be added instead. The ip address is taken to be the computer's ip address, but you can enter the port to run on and also the password and admin keys.

2.4 Start / Stop Server Group Box

The group box just below this then allows you to start the server running through the `Start Server` button. The `Passwords required` check box allows for the server to be password protected. By default, any service can ask the server (also a service) for its password, which will be returned. The passwords required feature will stop the server returning its password and so it can only be accessed if the password is already known. The default protocol is 'HTTP'. The `As https` check box will use 'HTTPS' at the start of the address instead, for the local server that gets started. This can add further security to the server interface, through the secure sockets that https provides.

After the server has started, you can then stop it through the `Stop Server` button. This is different to the admin option of clearing the network of all services but keeping the server running, which can be done through the `Admin` panel. Any log information produced from a locally running server can then be seen in the `Server` text area, as is shown in Figure 3. You can also remove a server from the view using the popup menu in the `Network` panel. In that case, it will still be running remotely and so if you enter its URL into the `Add Remote Server` box again and re-register it, the view should display the server with all of its network nodes again.

Note: The server log appears to show a lot of faults. The Warn message in particular can probably be ignored and some of the Exceptions. Sometime the loader cannot find the correct class or method first time, when it will throw an exception instead. More serious errors will often result in a MessageBox warning.

2.5 GUI and Service Configuration

The GUI can be configured during startup, to load or setup values that relate to the server and the services that run on it. This makes it easier to manage all of the config information that you might use. The information is saved in an XML file, details of which are included in the 'licasAdmin' document. You do not need to know the specific structure of the document however, where the details of this section describe only how to use the config form to setup the GUI. The GUI can be configured using a service factory. This can be done through a form that is opened using the service factory toolbar button, or menu option. The service factory configuration form might display configuration information as shown in Figure 4. Note that the default factory should load the very first time that you open this form and you should check this form the very first time you install the GUI. There are now two tab options. The first is for the service GUI configurations and the second is for creating a service admin document.

The top `Select Factory Module` group box displays all of the loaded jar modules. To add a new one, you need to click the `Service Module Manage` button to open the related form. Note that opening this form does not also load the config file into the GUI. It parses the file and displays the current settings, which you can then change and re-save. To load the config settings into the GUI, you need to use the 'Admin / Load GUI Config' menu option. The service factory can be used, for example, to define the forms-based GUI interface for a service, or add a new jar module with service classes. The `licas_services` package already contains the default service implementations with related GUI interfaces, but new services can be configured this way.

The form can therefore be used to declare a new type of service that can then be selected and loaded. The default classes also declare a set of standard services that are automatically loaded in during start-up. The hard-coded info, relating specifically to the default service classes, cannot be changed and so the details are not added to the script. If you add any of your own information to the script, you can then change it again afterwards. You can also enter new details for one of the existing service types. However, only one service class is allowed for each service type. So if you choose to overwrite an existing entry, you will be prompted, but it will be the new details that you enter that get used. If you then delete them again however, the original default service will still exist and can be used.

Every time you save the configuration, the current version will be backed up first. It is saved in a file with the same 'default_config' name, but with a '.bkup' file type name. Once a configuration has been created and saved, it can be reloaded using the 'Admin / Load GUI Config' menu option. The service factory itself is created as a static single instance and is initialised with the config settings. Note however that the configuration relates only to the local server, running on the GUI itself. If a remote server is registered, it cannot be configured through the GUI config files.

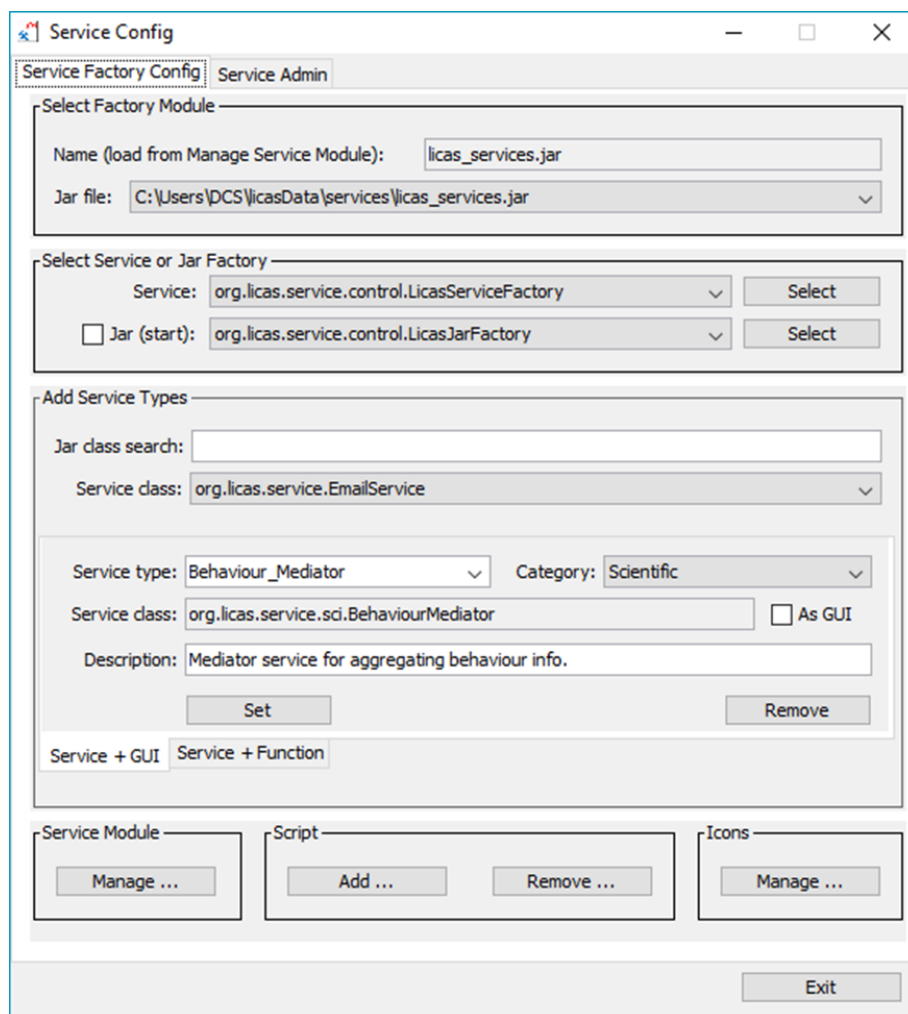


Figure 4. Service Configuration form.

2.5.1 GUI Admin Details

These details are not entered through the Service factory form, but are found on the main GUI itself. As you might want to change them, they can be saved to the config file as well. For a local server running on the GUI, you can save the server port and passwords to the config file. This might be useful if the passwords are quite cryptic and is done through the `Save Admin Details` button in the 'Local server and GUI' group box of the Admin panel. This will also automatically save the admin panel settings for the graphic configuration. When you load in the configuration again, if server and admin details have been saved, the components in the related group boxes will be updated to show those values.

2.5.2 Service Factory and Jar Factory

All of the class types that you might use are loaded from the list of jar module references, displayed in the top group box of Figure 4. The 'Select Service or Jar Factory' group box allows you to select from any module a service factory and a jar factory type. Only appropriate class types from the module are displayed in the respective lists. This is determined as follows: After selecting the module jar file, any classes that extend the `licas` base `ServiceFactory` class will be displayed as possible service factories. You can select one of those and click the `Select` button to make it the official service factory object. You will also have the option of saving the new details to the config file. This would mean that when you start the GUI again, you can simply read in the default configuration and any of these modules or classes will automatically be ready for use. For a completely new installation, you need to do this at least once, but the process is very easy and all options should automatically open at the correct place.

If there is no jar factory loaded, the second list allows one to be selected. After selecting the module jar file, any classes in it that implement the jar factory interface will be displayed. You can select one of those and click the `Select` button to make it the jar factory object. A jar factory is used to retrieve jar files from the local 'licasData/modules' folder and to parse them into XML for sending to a remote server. The remote server must also be running a jar factory that would then serialize the file object and save it in its own modules folder. The jar file is then used to create a new service with. This process is automated through the all-in-one GUI and uses the default server and service methods. There is also an option to disable or not start the jar factory. This would mean that retrieving remote jar files through this channel would not be available. This is done using the `Jar(start)` check box. Another option to completely disable loading remote jar files is to disable External Jars, described in section 3.2.

2.5.3 Specify new Service or GUI Interface

There is now an `Add Service Types` tabbed panel for adding details for either a new service type or function service. You can also specify a GUI interface for a particular service type. The top of this area displays a service class list for the currently selected jar file. It also has a search field, to allow you to filter the class list. Both of the add service options use the same class list. For adding a new service type, the first tabbed pane lists the current service types and the categories that they belong to. If a service class already exists for the type, it gets displayed, as does a description if one exists. The GUI interface class is a different class and so, if you click the `As GUI` check box, the related GUI interface class gets displayed instead. Selecting a different jar module will change the service classes list and it is the class in this list that is used for any service update. After deciding to add a new service type or GUI interface, click the `Set` button to save the association. The details are output again and you are asked to confirm the change. As only one service class is allowed for each service type, you might be overwriting an existing entry, but a subsequent deletion can reinstall the default setting again. You might also need to shut the application down completely and re-start it for all of the new details to load in again. The following default service GUI interfaces are currently available:

- **Message Service:** If this is not listed, select the `MessageServiceJFrame` class from the combo box and click `Set` to associate with the service.
- **File Service:** If this is not listed, select the `FileServiceJFrame` class from the combo box and click `Set` to associate with the service.
- **Web Service:** If this is not listed, select the `WebServiceJFrame` class from the combo box and click `Set` to associate with the service.
- **Behaviour Mediator Service:** If this is not listed, select the `BehaviourMediatorJFrame` class from the combo box and click `Set` to associate with the service.

The `Remove` button should remove the selected service type from the configuration instead. After saving to the config file, if the GUI is still not associated with the service, reload the config file through the `Admin / GUI Config` option and the association should be loaded.

2.5.4 Function Services Configuration

The last section allowed new default services to be defined. These are listed in the main All-in-One GUI panel service boxes, from where they can be selected and loaded onto a server. A number of other panels in the GUI allow you to select other classes for other functions. In particular, the `Scientific` panels would allow you to select a different type of service to

perform a different type of function, for example. The problem solving or behaviour scripts are therefore configurable with different service class types and other values. These are now declared in the configuration script and added through the service factory form. Note that a description section can be used to describe the relevance of each class type that is included. This is then displayed in the GUI, along with the class name.

The information is displayed in the second `Service + Function` tab. The selected class is chosen by the list of jar file classes in the top area. If you change the class there, it will update the selected class for this section as well. You then have the function type, category type and a list of the existing service classes for that function type. A function can be performed by different operations, represented by a different service class. Below that there is a text area for adding a description. This is shown in Figure 5. If the `Update` button is clicked, the class selected in the `Service class` list, would be added as the class for that particular function. After exiting the form, the details should be saved and so you can just check them again to see what has been added where. Additional function classes are not required with the default packages and so this would only be used if you were specifically adding something new.

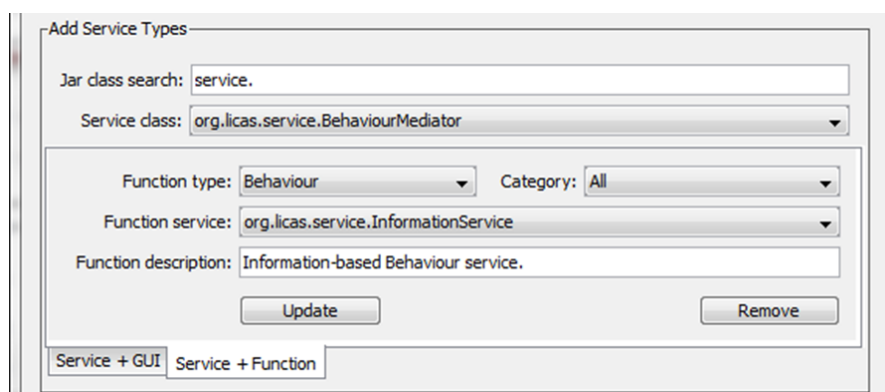


Figure 5. Add Function Service panel.

2.5.5 External Modules

One particularly attractive feature of this system is the ability to load in classes or services from external or third-party jar files. The path to the jar file needs to be specified, when its classes can then be loaded into the system. The `Service Module` option in the service factory form allows you to manage these external jar files. If the `Manage` button is clicked, a new form is opened that displays the currently selected external jar files, as shown in Figure 6. You can browse to `Add` a new jar file path to the list. This can then be saved, when

the list will be automatically loaded at startup time, meaning that the related classes will also automatically be available. After specifying a new module, for example, you can go to the `Service Panel` of the main GUI and add a new service to your list. You can then also select a file on the list and click the `Remove` button to remove it. Each jar file can be either removed completely and deleted, or you can deselect it using the check box. If it is deselected, then it will not be loaded at startup time, but the file reference will not be removed. Any new jar module that is added is also displayed in the main `Service Config` form, so that you can use its classes to configure the GUI setup.

After changing any of the `Include` check box selections, you need to click the `Update` button to save the new settings. If you delete a module completely, then if there are GUI interfaces or default services associated with it, they will be permanently deleted as well. You will need to enter those details again if you then add the module again. Therefore, the option to not include might be preferred, when the other information will not be deleted. When referencing an external jar file, you have the option of copying it to the default 'licasData/modules' folder first. If you do that, then it will always be available and there is no danger of it being deleted. You can alternatively reference it from where it is. In that case, if you update it, the update will also be available to the licas system. There may be problems reading some jar files properly and so you might receive a warning about that. It is also important to compile the jar into a single file, as moving it would make any third-party jars that it uses unavailable.

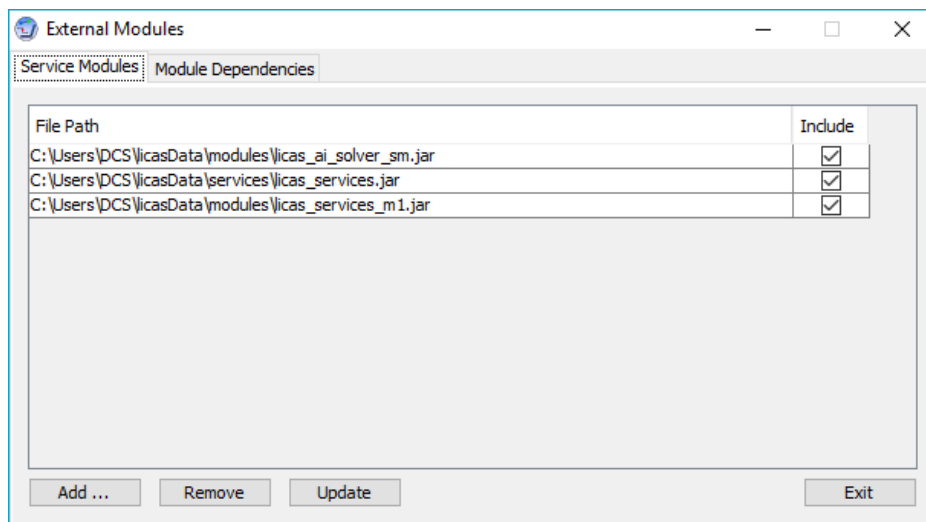


Figure 6. Jar Modules form.

When loading in the main jar, the system looks for a 'lib' folder with dependency jar files. If they are present, you are asked to load them in as well. The second `Module Dependencies` tab will then list the jar files that the main one requires.

Note: the jar file reference is stored under the jar file name. It would be 'licas_test_objects.jar' in Figure 6. You should not try to overwrite an existing jar file. It might be the case that it is the same file that you are trying to copy. If you delete something and the classes are not automatically updated, exit the config form and open it again, when the configuration should then be correct.

2.5.6 Service Icons

It is also possible to represent a service with a specific icon on the graphic. These are again hard-coded for the default services, but they can be configured for any other services that you might provide. The `Icons` option in the service factory form allows you to manage these external jar files. If the `Manage` button is clicked, a new form is opened that displays the currently selected external jar files, as shown in Figure 7. You can add or remove icons from the default resources folder that is in the directory that your application starts from. These should be of a size 25 x 25 pixels, but can be of most graphic types – png, gif, etc. If you then click a row in the table, representing one of the external module services, you can change the icon to display for it, or reset to the default red rectangle that is displayed when there is no icon.

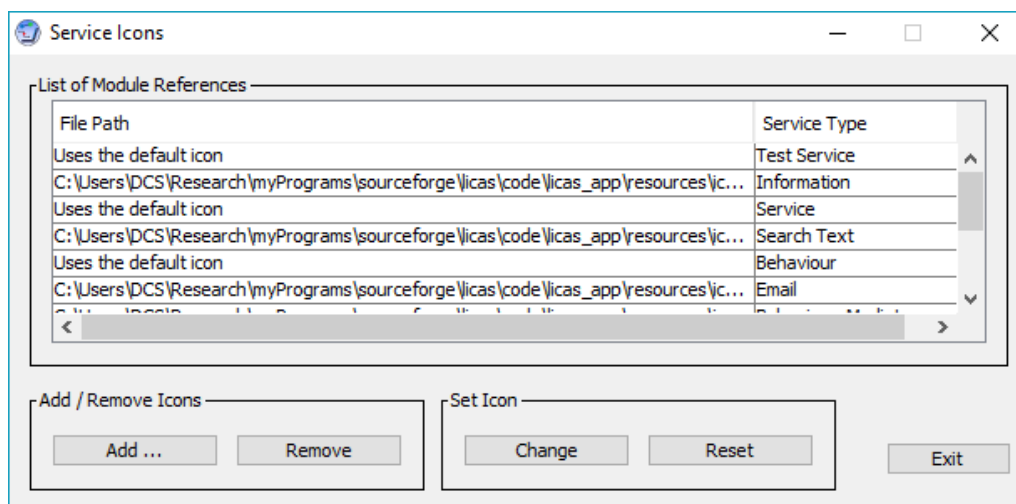


Figure 7. Icons Form

2.5.7 Adding New Default Services

The `Service Panel` displays a number of default services that come with the base `licas` package. If you load in your own module, you can then select classes from it using the `Service panel` boxes and add them as new services that can then be loaded onto a server. When you add a new service definition, you will be asked if you want to save the details permanently in the configuration file. If you confirm, then when you load in the service factory and related modules, these service definitions will also automatically be loaded, allowing you to use them without needing to enter the details again.

Note: To change the default service list, for example, to add a different messenger service, you would need to change the service factory itself, as it hard-codes the default service info that it can provide. Selecting the new service factory as the official one should then also update the service details.

2.5.8 Copying Scripts

As a convenience, it is also possible to copy an XML script file to a default location, so that they are grouped together for easy access. The GUI automatically opens at this default location when a script is required. The `Script` option in the service factory form allows you to add or remove scripts.

2.6 Service Admin Document

The second tab in the service factory form allows you to browse over the class data and create an admin document for initialising a service with. This should include the fields described in the 'licasAdmin' guide. You can set basic descriptions or the `Auto` class loop times, or change the autonomic manager to use, for example. The document can be saved and used to initialise a service with. The easiest way would be to enter values from the form, to see what type of document is created.

2.7 Security and Web Manager

As the GUI can now be used as a web server, to retrieve files directly to a browser, a basic security manager has been added. It can currently only configure folder or directory locations that are specified as legal or accessible from the Internet. The security settings are available from the main toolbar menu, where Figure 8 shows the configuration form. If you are unsure, you do not need to change anything, when there is always the default location for adding your files. For the web manager, there is a default 'web' folder that gets created

in the root 'DCS' folder of the installation. This is always present and accessible. You then have the option to browse to any other folder and add it. For the licas system, any location typed into the browser gets added to these default folders and if the file is present, it can be returned. Therefore, any location allows any file or sub-folder to have the same access rights.

The HTTP address is then the server HTTP plus the additional path from a default location. So, for example:

- `http://192.168.2.2:8080/testHtml/default.html` might access a folder 'testHtml' in the default 'C:\DCS\web' location and return the file `C:\DCS\web\testHtml\default.html`.
- `http://192.168.2.2:8080/paper.pdf` might access a pdf file in some location. If for example, you add your documents folder to the list (C:\Users\You\Documents) it can look for a file called `C:\Users\You\Documents\paper.pdf`.

After adding or removing anything, you still need to *Save* to make the action permanent. The logger can also be configured where changes are updated dynamically. The logging channels are now stored centrally and if you change the config settings and 'Save' again, a new config file gets written and re-loaded. It should not create any existing channel again, but might update some of its settings. New default channels can also be added – a file, for example.

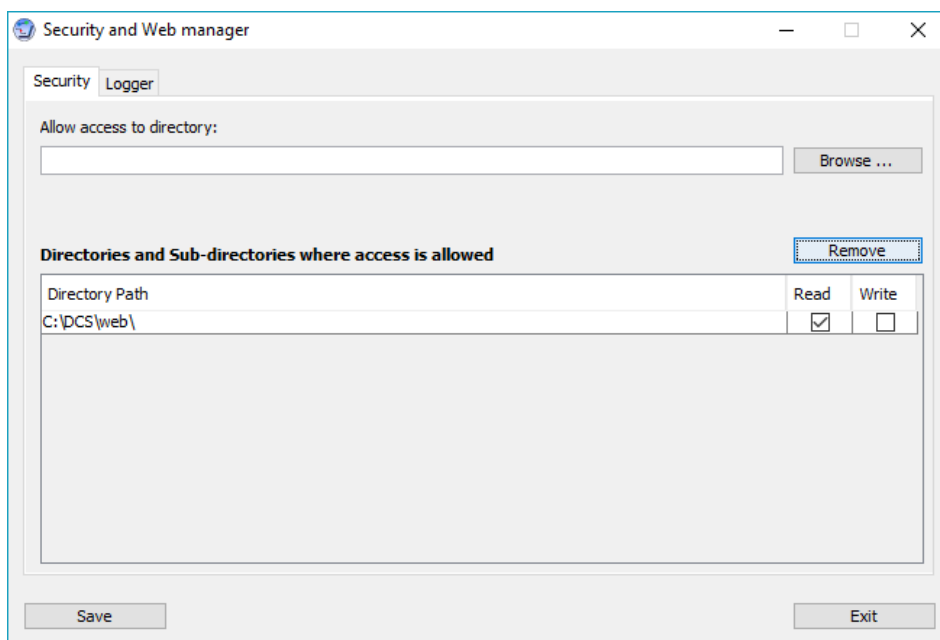


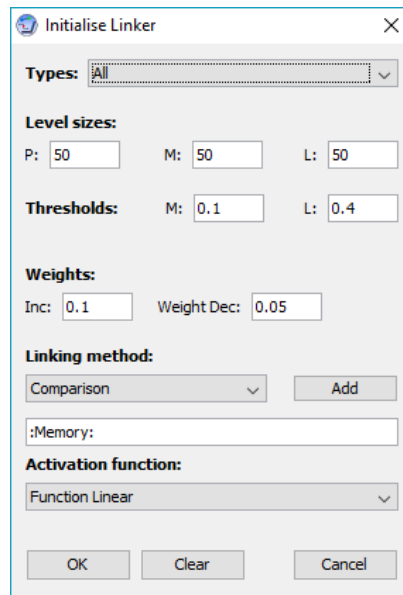
Figure 8. Security and Web Manager with default settings.

2.8 Server Popup Menu

Each service that is loaded has its own popup menu that allows some operations to be performed. The server in the Server graphic also has a popup menu that allows some server-specific or service-general operations to be performed. For example, default utility services can be added to all running services. The menu options here are as follows:

- **Server:** Options here allow you to add or remove the jar factory loader. This could become a security risk and so it is easy to remove it using this menu option and then add one again, using the loaded config settings.
- **Metadata:** Some general metadata is displayed when this option is selected.
- **Utility Service:** there are two services defined as utility services. These are the dynamic linker and the timer services. Utility services are intended to perform some function for another service, but are general in their application. You can add them to all running services, or a selection of them. A linker service, for example, can be added to a specific service, service type or all of the loaded services. A timer service can similarly be added.

The linker utility service can also be initialised through its own form that is displayed in Figure 9. A set of default values are available when the form is loaded, but you can change these to anything else. The values define the requirements for the linking mechanism and structure, with the 3 levels and 2 thresholds. The `Types` option at the top can be used to select all base services, services of a particular type, or a single service with a specific uuid. The linking mechanism itself can also be defined, although not all options might automatically work. If in doubt, the default `Memory` one should just be used. This linker form is also opened by the scientific panels, to allow you to enter linking configuration values into a script. The threshold value is now also passed through a function first. There is a default linear one that returns the same value that is input, or a Sine function. If in doubt, stick with the default linear one.



The image shows a dialog box titled "Initialise Linker" with a close button (X) in the top right corner. The dialog is organized into several sections:

- Types:** A dropdown menu currently showing "All".
- Level sizes:** Three input fields labeled "P:", "M:", and "L:", each containing the value "50".
- Thresholds:** Two input fields labeled "M:" and "L:", containing the values "0.1" and "0.4" respectively.
- Weights:** Two input fields labeled "Inc:" and "Weight Dec:", containing the values "0.1" and "0.05" respectively.
- Linking method:** A dropdown menu showing "Comparison" and an "Add" button to its right.
- :Memory:** An empty text input field.
- Activation function:** A dropdown menu showing "Function Linear".

At the bottom of the dialog, there are three buttons: "OK", "Clear", and "Cancel".

Figure 9. Linker Service initialisation form.

3 Admin Panel

Figure 10 shows the admin panel. This can be used to change or configure the network display and also allows the user to view other information relating to the running network.

3.1 Server Admin Group Box

There is a server admin group box for performing admin operations on a network as a whole. To retrieve the different service types from the network you can click the `Get Service Types` button. Then you can select a service type and click the `Start Threads` button to start all services of that type running on the network. There is also an `All Services` option to start all service threads of any type.

Each synchronous call made on the network is allowed only a certain amount of time to execute. Synchronous calls make the call and then wait for a reply, and so to stop a call from waiting forever and therefore blocking any other operations, there is a timeout value associated with each call. The timeout value is defined in milliseconds, or thousands of a second. The default value is 10000 milliseconds, or 10 seconds. If you think that your call will take longer than this, then you can set a new timeout value in the `Sync call timeout` text box. If a numerical number is entered here, then this will be converted into milliseconds and used as the timeout value for the call. If the timeout value is reached before the call completes, you will receive an error message in the GUI.

You can also clear the network running on a server, or delete all services running there. To do this, click the `Clear Network` button. Note that this does not shut the network down, but only removes all of the currently running services. So you could clear the network and then load a new network onto the same server afterwards.

3.2 Allow / Block Add Services

As the server is open once the password is known, this is an override option that allows you to block any more services from being added. You can click the `Block add service` radio button to force blocking of any more services from being added. You can then re-allow services to be added by clicking on `Allow add service` button.

There is also an `External Loaders` option to block any external jar files being used as classloaders. The default is to have this box checked, which means to allow external or

remote classloaders, but you can uncheck it and save the config to not allow any external classloaders.

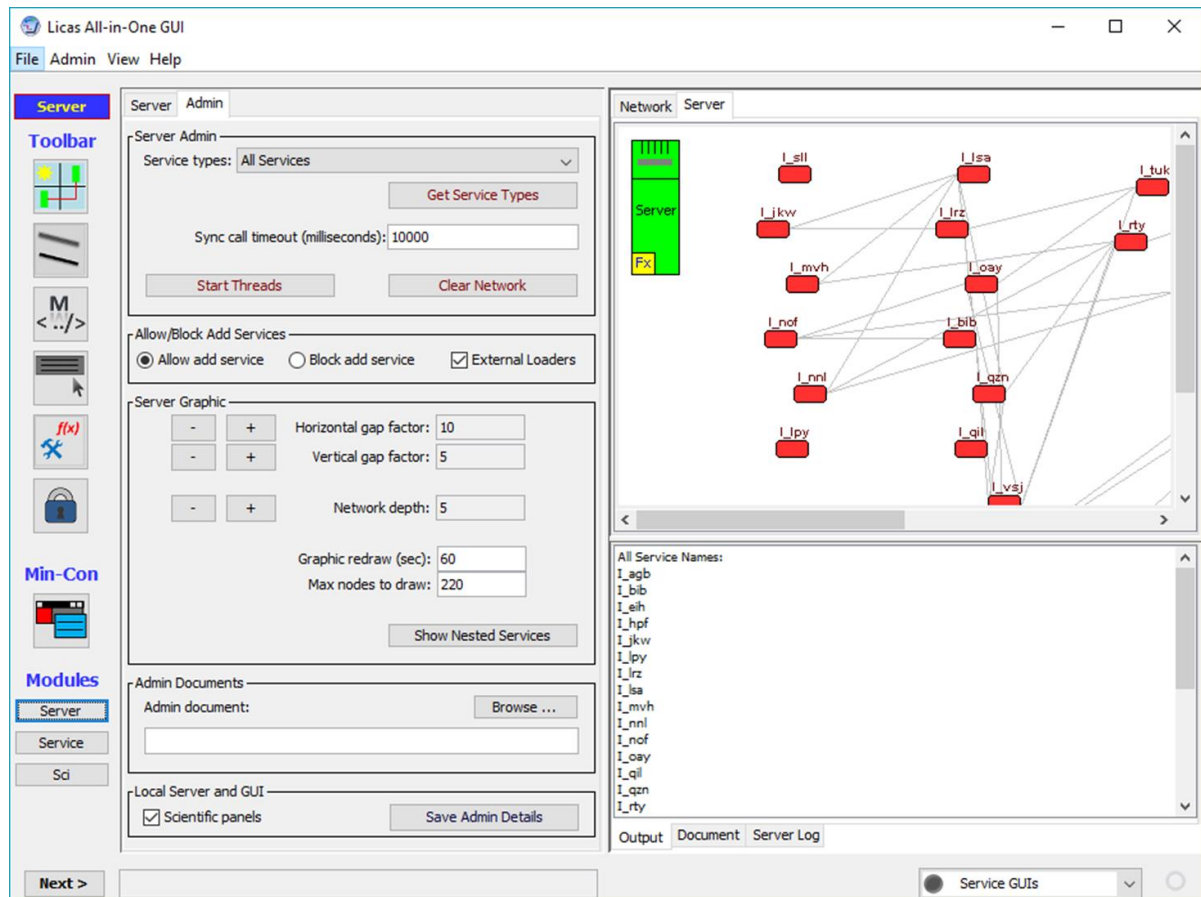


Figure 10: Admin Panel

3.3 Server Graphic Group Box

This group box contains some general options for adjusting the server display. Values for the vertical and horizontal spacing between components are displayed. These values are adjusted by clicking the increment or decrement buttons related to them:

- The width between components can be enlarged/decreased by clicking the Horizontal gap +/- buttons.
- The height between components can be enlarged/decreased by clicking the Vertical gap +/- buttons.

In the figure, for example, there are lots of links from one service to another. The vertical gap has been increased to make these easier to see. With the short service names, the horizontal gap has also been decreased.

You can also do the same thing with the `Network depth` option. This can be used to limit the depth that a network of links is drawn to. By default, the network will be drawn to a depth of 5 permanent links in a hierarchical structure. Changing this value will adjust the depth accordingly. You can then click the `Refresh View` toolbar button after any change to update it on the graphic. There are also options to adjust the size of network that is displayed and the refresh time. These are determined by the related editable text boxes:

- The `Graphic Redraw` time changes the time between which the network graphic is automatically redrawn. This is entered manually and changed by adjusting the value in the text box. The time is entered in units of seconds and defaults to 60 seconds. If you ever want an immediate update of the network view, you can also use the toolbar `Refresh View` button.
- The `Max Nodes` text box determines the maximum number of nodes that should be drawn, regardless of the maximum depth specified in the network depth box. So if there are additional nodes at an allowed depth, but the maximum number has been reached, the graphic will stop drawing new nodes.

By default only the base services are displayed, but there is a toggle button called `Show / Hide Nested Services`, which when clicked, can show or hide the child services as well. The starting position is to hide the nested services and show only the top-level services. If you click the toggle button, you can show all of the nested services. You can then click it again to hide the nested services, and so on.

The 'network redraw and size' options have been added because there might be memory or threading problems when drawing the graphic. This can be related to the total number of nodes that it tries to draw and also the number of times it tries to redraw them. So these two parameters have been made editable and can be adjusted by the user. This is not a fool-proof solution but should make it practical to display at least parts of larger networks as well.

3.4 Enable Scientific Panels

As the All-in-One GUI would not normally use these panels, they have been set to be hidden as default and so this is an important option, to show the scientific panels. There is a check box beside the save admin button. You need to check this and then save the config file as the default config. You then close and re-open the GUI to show the scientific panels.

3.5 Save GUI Config

If you click the `Save Admin Details` button in the 'Local server and GUI' group box, this will save both the server port/password/admin key and the graphic configuration settings of the admin panel to the default config file. You can save this to the default config file `default_config`. When you load in the configuration again, if server and admin details have been saved, the components in the related group boxes will be updated to show those values.

4 Service Panel

There are three panels related to loading services. The `Service` panel can be used to load and run service types, or manually add new services. The services are stored under categories. The list of categories is hard-coded, but you can change what services belong to each category. There is only one allowed service class saved for each service type, but you can easily make up a similar type, for example. Figure 11 shows what the service panel looks like. When the service factory (section 2.5.2) is loaded, it adds the default service list from the config file. All relevant information is saved and so you can select one of these default services to run on the server. You can then add your own jar files as additional external ones. This is done by adding a new jar file module (section 2.5.5). If you want to view the metadata for any default service type, you can click the `Display Metadata` button and it will be shown in the output text area.

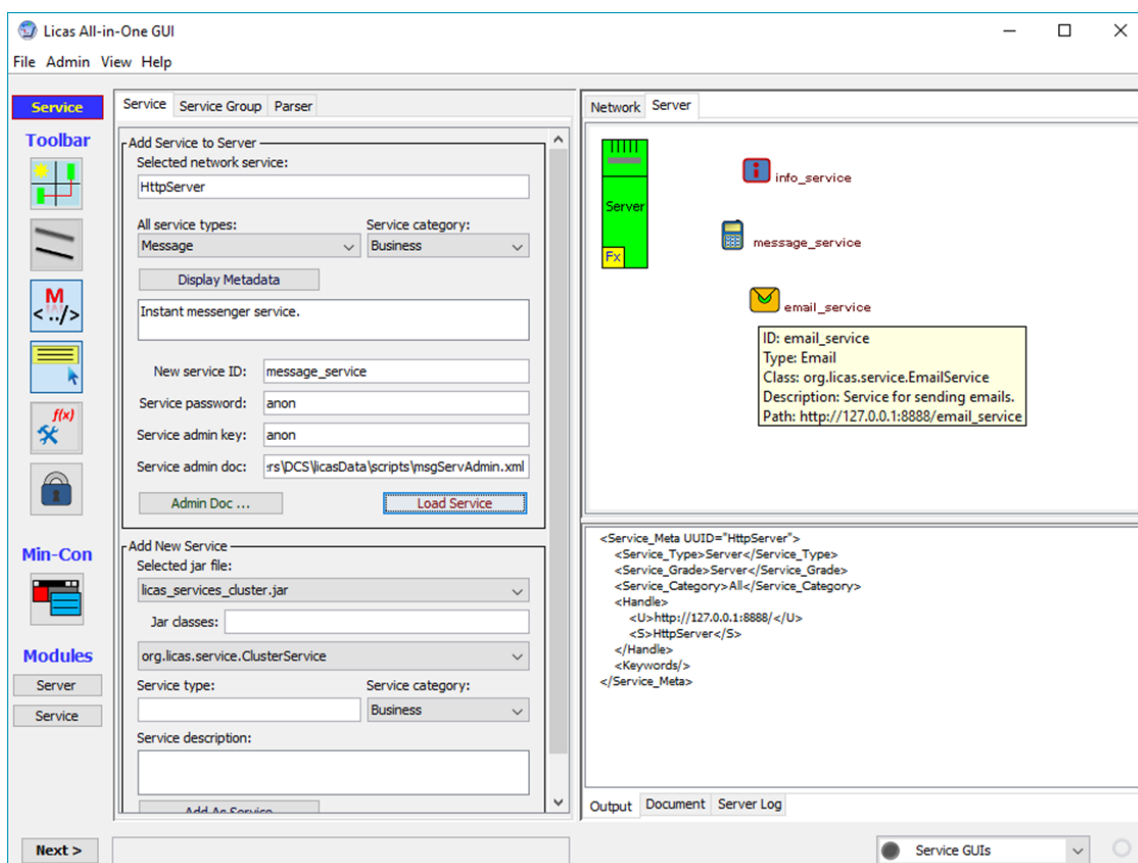


Figure 11: Service Panel

4.1 Add Service to Server Group Box

To add a service to the server you need to select the service type **(1)** from the `All service types` combo box in this group box. After selecting the service, you need to define where it should be loaded. This is typically directly on the server, or as a child service to another service, already loaded on the server. On the server graphic panel, you need to click one of the components. That component's ID will then be displayed in the `Selected network service` text box. This is the service that the new service will then be added to. You would normally click the server **(2)** itself, as utility services can also be added through the popup menus. After selecting the service type **(1)** and the service to add it to **(2)**, you can simply click the `Load Service` button to add the service. If you then refresh the view a couple of times, it should be displayed with the appropriate icon.

4.1.1 Service Constructor

The constructor for the service is now assumed to be the default for the Service class. There is now therefore a set list of text boxes for entering the information. The service password and admin key are set to `anon` by default, but can be changed to anything. You then need to enter the service name at least. This is typically enough, but if you want to add an admin script document, then you can browse to it using the `Admin Doc` button, when the selected file path should be displayed in the fourth text box. You can also manually remove it from there. If ignored, a service is loaded without any admin script, which is usually the case.

The `licas` web site and other documentation give more information on loading in parameter objects from XML-based files. If you then click the `Load Service` button, the service should be loaded onto the server and located with the correct parent. The network is redrawn periodically, but the `Refresh` toolbar button can update it immediately. Figure 11 shows three services that have been loaded. The mouse has also been moved over the graphic component for the same service and its tooltip info is displayed. Figure 14 possibly shows the constructor process in more detail.

4.2 Add New Service Group Box

The bottom set of group boxes allows you to add new service descriptions manually through this panel. They can then be loaded, or saved as part of the default list. You can use the search box to type in part of a name, when only the classnames with that text in their description will be displayed. To add a new service description, you need to specify all of the information that the automatic service factory would require. The first piece of information is the service class name. The top combo box displays the available jar files, and below this are the classes in that jar file. If you change the jar file you will then get a different set of

classes. The only place to load in a new jar file now is as a module through the `Service Factory` form. All of the loaded jar files however, are listed all related panels. There are then two entries, one for the service type and one for its group category. The categories are pre-defined and so you just select one to add it to. The 'service type' is free text and it should probably be a new type, as only one service class is stored for each default service type. There is then another text box for a metadata 'description'. You need to fill in all of this information manually and then click the `Add As Service` button. This will transfer the service information to the `Add Service to Server` group box, where the service type is displayed in the `All service types` combo box. See section 6 for a better example. The `Start Services` button will start the threads of all loaded services.

Note: Not all jar files from the Internet, for example, can be automatically read and parsed.

4.3 Use the Service Factory to define a new Service

You can also declare new services through the `Service Factory`, which is probably the best way now. Figure 12 gives one example using the default `licas_service` package.

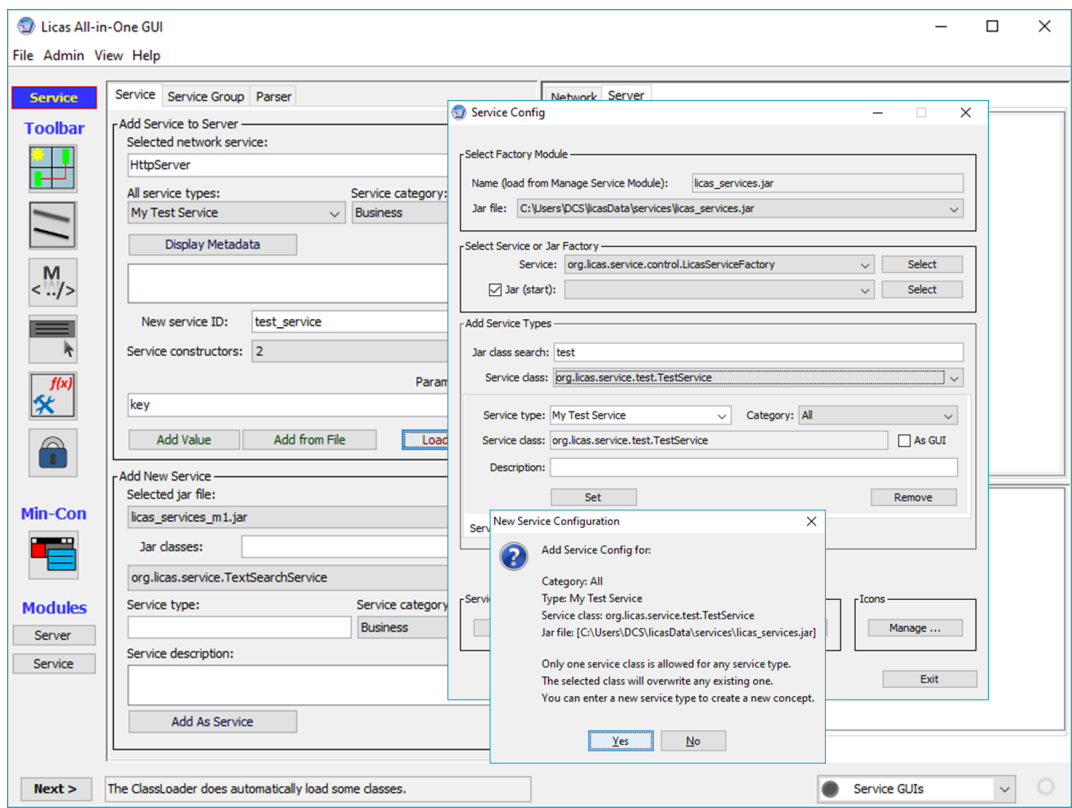


Figure 12. Test service from the `licas_services` package declared in the `Service Factory`.

You can list that package, as shown, by selecting it from the top `Jar file` combo box list in the `Service Factory` form. The class selected class (Service and Gui tab) is then the `TestService`. You can select a category type and also need to enter a new service type. This might be 'Test Service', for example and you can also enter a description. You then click the `Set` button to add it as a new service type. You will receive a confirmation message that you have to click OK on to then save the new service details. These are saved to your default GUI config file. You then close the GUI and re-start it. If you go to the `Service` panel, the new service should be included. You can select it and load it in, as any other service. In this case, the service requires constructor parameters, so you would need to select a constructor and enter the number of parameters required. The parameter values that were entered are shown at the bottom of the Output window.

4.4 Popup Menu Options

If you right-click on one of the service components on the graphic, a menu will pop up with several options related to that component. There is also a popup menu for the server. The service options are as follows:

- **GUI:** If there is a GUI interface associated with the service, selecting this option will open up the interface and link it to the service.
- **Serialize and save service:** This creates a full XML-based description of the service automatically, instead of the typical serialized byte code. It is still a work-in-progress but it can save certain parts already. Its usefulness is the fact that it can be configured when you load the service, to save only certain parts, so only certain parts would be sent over a network, for example, to be re-constructed somewhere else. A complete solution would be very useful, but resources are limited.
- **Metadata:** The metadata that is displayed when a service is clicked has changed slightly, but is still a subset of all of the available metadata. This menu has an option for each of the other metadata types, including the data section itself. So it is easier to check on the internal contents of a service now.
- **Utility Service:** there are now three services defined as utility services. These are the dynamic linker, email and the timer services. Utility services are intended to perform some function for another service, but are general in their application. You can add them at the service-specific level, to only one service. If you click on the server however, you have the option of adding them to all loaded services or a particular service type. The Linker service then opens another form that allows you to configure it properly. Unless you are performing serious tests, you do not need to worry too much about the linking method, but memory at least should probably be included.
- **Remove Service:** This option allows you to remove or delete the service from the network. Any links from other services to this one should also be removed.

- **Add Permanent Link:** You can manually create or destroy permanent links between services. At the moment, this can only be done between the base services coloured in red, which will probably now be the only available option. To create a permanent link, select this option. The mouse cursor will then change to a cross, when you then need to also click the service to link to. When you click that, you should receive a message stating that the link has been properly constructed and it will then be displayed on the network graphic.
- **Remove Permanent Link:** You can delete a link by following the same procedure, but selecting this option instead.
- **Get dynamic Links:** The network graphic only displays the permanent links. You can however retrieve the dynamic linking information by choosing this option, which will retrieve all of the dynamic links that are at the link level and display the service paths only.
- **Nudge:** This option allows you to nudge the component in any of the four directions by a small amount. This amount will remain permanent and will allow you to move it to a slightly better position for viewing, if the default graphic is not perfect. Simply select the Nudge option and from this select Up, Down, Left or Right.

5 Service Group Panel

This panel can be used to load in a group of services instead of just one service at a time. It is possibly most useful with the information service and works in a manner that is a combination of the `Service` panel and the `Behaviour` panel described later. Figure 13 shows what the Service Group panel looks like. Note that the service types to choose from are displayed in a single list, without the categories, but all of them should still be available.

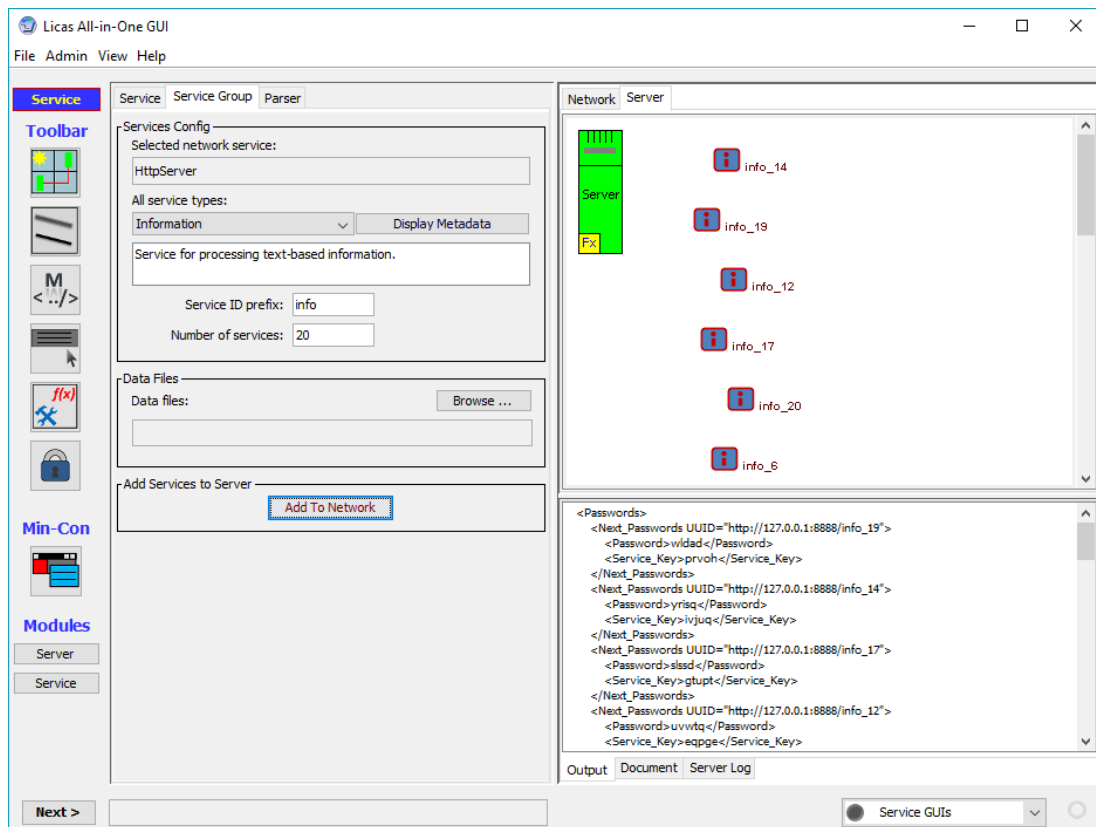


Figure 13. Service Group Panel.

As with the problem-solving panels, there is an option to browse to a folder that contains data files. Two default ones are provided with the installation. The services are always loaded onto the server only. You need to select the service type and specify the Service ID prefix. You also enter the number of services to create, but this gets overridden by the number of files if a folder is specified. The data files are simply read as is and added as data elements. There is no parsing to any data object. To declare a data type as well, you can use the behaviour panel with a limited specification and do not specify to automatically start the services running.

6 Parser Panel

The parser panel can be used to load parsers onto the network. Web Services can typically handle simple types only – String, Integer, Float, etc., so this is a problem if one of your parameters is a complex object. XML-RPC can use parsers to parse known object types to or from XML, allowing them to be passed remotely as well. The licas system however already provides a lot of default parsers and so it may be quite rare that you would need to add any of your own. If you do, then from this panel, a user can select the parser class and the class to be parsed and add them either locally or to any of the registered servers. This is illustrated in Figure 14.

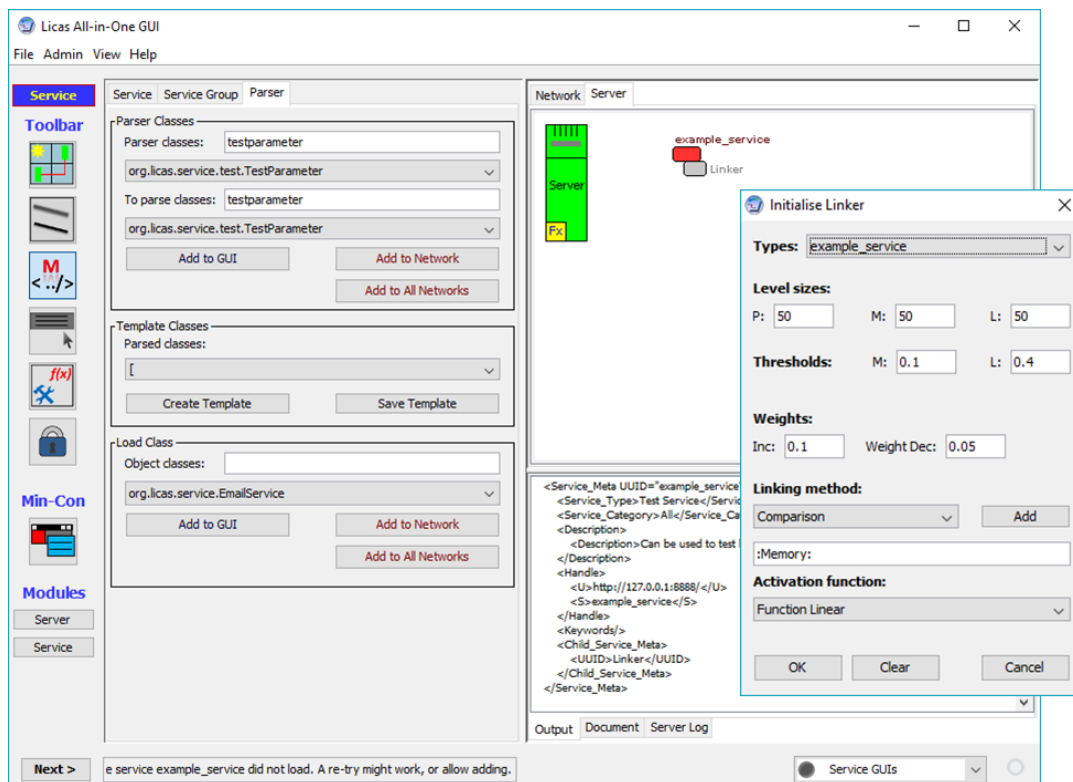


Figure 14: Parser Panel

A Test service has been loaded, using the default constructor that requires a password and an admin key. Using the test service’s ‘Menu-Add Utility service’ option, the Linker service has been selected and added, which automatically opens the Linker config form. In the Service panel, make sure that the `licas_services` jar file is selected. The `TestService` class of the default ‘licas_services’ package contain one method that

requires a `TestParameter` object. This is parsed by the `TestParameterParser` parser, as follows:

6.1 Parser Classes Group Box

This group box can be used to add a parser to the system. There are two combo boxes that display the classes for the currently selected jar file. This is the same jar file that is selected in the `Load Service` panel. You can select the parser class and then also the class of the object that is parsed by that parser. Note that the parser will only be added if it implements the `ParserDef` interface. If this interface is not implemented then the parsing process will not be able to automatically invoke the appropriate parsing methods.

When the correct classes are selected, they should firstly be added to the GUI by clicking the `Add to GUI` button. They should then also be added to the network. To add to the network that is currently displayed, click the `Add to Network` button. To add to all registered networks, click the `Add to All Networks` button. Figure 14 shows that the parser to be added is called `TestParameterParser` and it will parse a class called `TestParameter`. When adding to the GUI, you also have the option to save the parser description in the config form. If you do this, when you load in the config form again, the parser will automatically be created and added to the default set.

6.2 Template Classes Group Box

This group box can be used to create a template that can be used to construct an XML-based representation of a complex object. This object can then be used as a parameter, as part of a method call. A combo box here displays the parsers that are currently loaded, or rather, the objects that they parse. If you click the `Create Template` button, the parser will be used to create an empty XML-based representation of the parsed object. This is an XML-based description with an additional keyword at the start – to identify its type to the licas system. This process is also dependent on what the parser produces, so when you add your own parser, the template will only produce what the parser itself provides from an empty class.

The template is displayed in the `Document` text area, which is editable. What you can then do is update the template with specific values and then save the template to a file by clicking the `Save Template` button or using the main menu. The file needs to be a text file (.txt) because the format will not be recognised as XML. If you then execute a method that uses the object, you can load in the file path as the parameter value and it will be read

and parsed into the object before the method is called. So this provides some sort of manual process for testing methods with complex parameters.

6.3 Load Class Group Box

You can also try to load a classtype into the JVM, to check if it is possible. Note that loading a class also adds the whole jar file loader to the list of loaders and so it may not be necessary to try to load any of the other classes afterwards.

7 Method Panel

The method panel can be used to execute any method on any service in the network. Figure 15 shows what the method panel looks like. It can be opened from the new Utility menu option and panel set. This example shows the earlier information services. One of the services – User1_1 has been selected and the GET method has also been selected, which does not require any parameters. As they stored Bag-Of-Words objects, that has been returned as the service’s information.

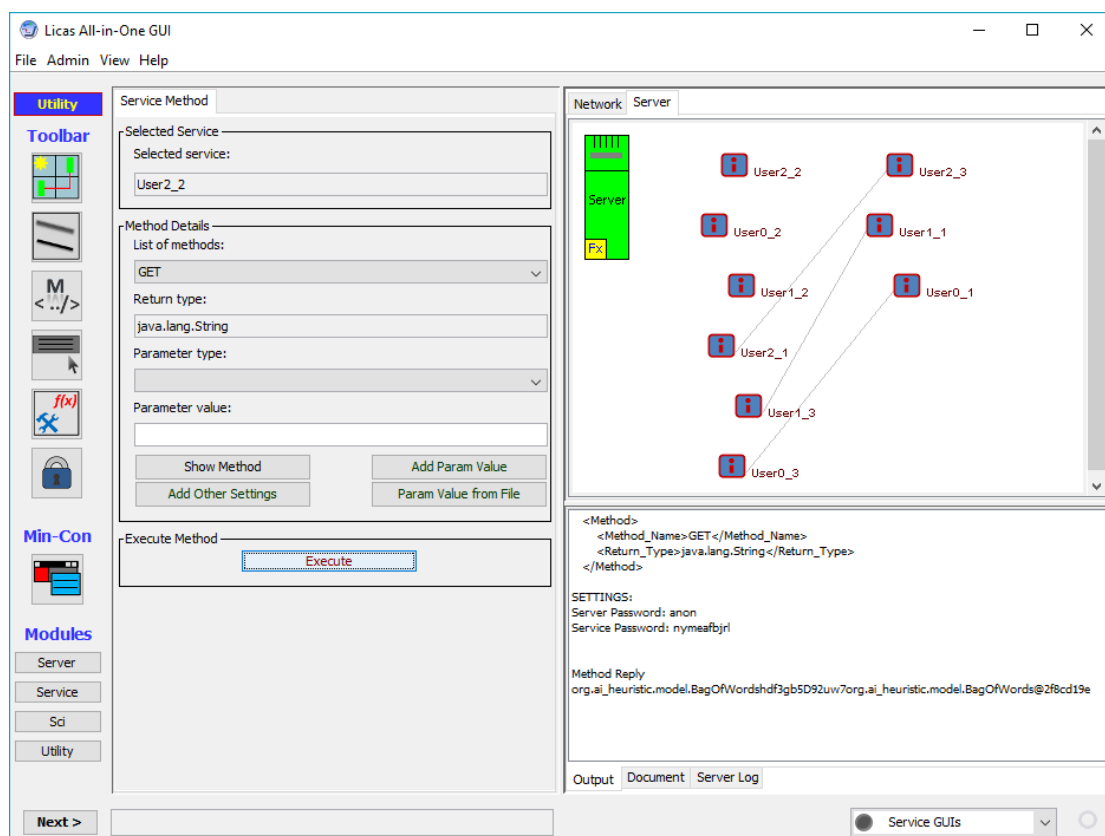


Figure 15: Method Panel

7.1 Selected Service Group Box

This group box provides details of the service that the method will be executed on. To select a service you should click on it in the network graphic. Then the service name should appear

in this box. The methods that are available in that service are then retrieved and displayed in the method details box, described next.

7.2 Method Details Group Box

This group box provides details of the methods that can be executed on the selected service. The `Methods` combo box provides a list of all available methods. There can be a number of methods, each with the same name but different parameter specifications. In this case additional methods with the same name will have a number after them, but this is only to tell them apart in the combo box. When they are executed, the correct name will be used. The `Return Type` text box then displays the return type, while the `Parameter Type` combo box displays the parameter types for the method, if there are any.

If there are parameters, then you need to enter values for each one. If it is a textual value, you can enter it into the `Parameter Value` text box and then click the `Add Parameter` button. If the parameter type is more complex, then it needs to have been stored as an XML-based text document previously, as described in section 5. You can then load in this value by clicking the `Xml Parameter from File` button and loading in the text file. The textual description of the parameter will then be displayed in the output area under the parameters section. Note that the parameter types are numbered and relate to each specific parameter. If you enter a value for a parameter with the value of (1), it will be stored for that parameter only. If you enter a new value and do not change the parameter number, the stored value will simply be overwritten. The parameter should however move onto the next one every time you enter a value, but you should keep an eye on this.

If the parameter is a `Vector`, then it is possible to add more than one value to the parameter and it will be stored as part of a `Vector` object. Simply select the relevant parameter type and enter a value and click the `Add Parameter` button more than once. In that case, you may need to reset the parameter number in the combo box before adding more values. After all of the parameters have been entered, the method can be executed.

7.2.1 Other Method Info Form

If you click the `Other Method Info` button on the `Method` panel, this will open another form that allows you to enter a password, communication ID, or packet size. Figure 16 shows what this form looks like. If you enter values into this form and click the `OK` button, they should be displayed with the other method details in the main GUI before the method is executed. The GUI should already know what the password for the service to invoke is, so the entry field here can be used to enter an incorrect password for testing

purposes. This should then replace the password that would automatically be used. The communication ID is useful for autonomous communications, while the packet size allows you to test sending a message in parts instead of a single transaction. The packet size is measured in bytes and requires a communication ID in all cases.

The image shows a dialog box titled "Other Method Info" with a close button (X) in the top right corner. Inside the dialog, there are three text input fields stacked vertically. The first is labeled "Change service password:", the second is labeled "Communication ID:", and the third is labeled "Packet size:". At the bottom of the dialog, there are two buttons: "OK" on the left and "Cancel" on the right.

Figure 16: Other Method Info Form

7.3 Execute Method Group Box

This group box allows you to execute the method that has been selected and configured. There is also a `Show Details` button that when clicked will show the details of the currently selected method. The `Execute` button then executes the method. Note that the service password used is the admin one and so this should allow access to any method. This is shown as part of the method details that are displayed.

When you execute the method, any complex parameter objects that are represented as XML-based text will be automatically parsed back into Java Objects and then passed to the method in this form. So this would be the same as creating these objects as part of Java code and then using them as part of a method invocation. The reply from the method call will also be parsed, but this time from the object back into XML and then displayed in the output text area in this form. More details on this can be found in the following sections, on the web site, or other documentation (user guide).