



Weave

Business Integration Framework

Weave System Administrator Guide Excerpt Feb 2022

- 1. Installation Guide 3
 - 1.1 Server Software Requirements 3
 - 1.2 Server Hardware Requirements 4
 - 1.3 Client Software Requirements 4
 - 1.4 Supported Platforms 4
 - 1.5 Example Size and Hardware Specifications 5
 - 1.6 Installation 5
 - 1.6.1 Console and Automated Installation 9
 - 1.7 Upgrading 10
 - 1.7.1 Upgrading from 2.4 to 2.5 10
 - 1.7.1.1 Network Ports 22
 - 1.7.2 Upgrading from 2.5 to 2.6 23
 - 1.8 Latest Downloads 31
 - 1.9 End of Support Announcements for Weave 32
 - 1.10 Weave Hub - Third-party Application Integration 32
 - 1.10.1 What is the Weave Hub 39

Installation Guide

Weave is a *web application*, meaning that it runs centrally on a server and users interact with it through web browsers from any computer connected to the intranet or internet.

The software platforms currently supported for Weave are listed on the [Supported Platforms](#) page. More detailed information regarding the software and hardware requirements of Weave can be found on the [Server Software Requirements](#) and [Server Hardware Requirements](#) pages.

The pages also cover information about:

- Platforms which Cohga does not support, but should work with Weave.
- Platforms and other software that are incompatible with Weave or have known problems running in conjunction with Weave.

Server Software Requirements

This page describes the server software requirements for Weave.

Java

The Weave web application requires Java to run, but as part of the installation process a suitable Java runtime can be installed along with the rest of the Weave components. During the installation process you can choose to use an existing Java runtime for running Weave, if it meets the minimum requirements, or you can have the Weave installer install the bundled Java runtime. The Java runtime that's bundled with the release is 1.7 and is suitable for both Windows and Linux, but Weave will also work with Java 1.8.

If however you are trying to install Weave directly from the installation file or you've copied the installer to a remote server that you don't have hardware access to, then you'll need to, at least temporarily, install a Java runtime on the server. The Java runtime used for the installer does not have the same requirements as the Java runtime required to run Weave itself, so an older Java installation may be suitable.

We do not provide support for non-Oracle Java versions. However, if you wish to install a non-Oracle Java and you want to use SSL, you will also need to install the [Sun JSSE package](#).

OpenJDK is currently not supported.

Java Licencing.

Cohga supplies version 7 of Java with Weave by default. This supplied version of Java 7 does not have updates enabled and is fixed at the version, as installed. Cohga has supplied Java 8 to some customers that need Java 8 to work with Esri using TLS 1.2 and above for their ArcGIS Online platform. The Java 8 version supplied by Cohga is build 191 and was downloaded in September 2018.

The Oracle JDK FAQ clearly states that if you download and use a version of Java prior to April 16, 2019, it is not affected by the new license agreement. See here: <https://www.oracle.com/technetwork/java/javase/overview/oracle-jdk-faqs.html>

Specifically these two sections:

What versions are available under the Binary Code License for Java SE (“BCL”)?

Oracle Java SE versions (including updates) released prior to April 16, 2019 are the only versions licensed under the BCL license.

What happens to the Oracle Java SE releases I downloaded under previous licenses, such as the BCL?

You may continue to use releases you have downloaded under the terms of the license under which you downloaded them. Legacy releases are still available in the [Java Archives](#). Note that older versions of the JRE and JDK are provided to help developers debug issues in older systems. They are not updated with the latest security patches and are not recommended for use in production. Oracle strongly recommends that you remain on an up-to-date version of Java with the latest performance, stability and security updates.

Importantly, *later* releases of Weave will use the OpenJDK version of Java. This does not have the licence restrictions as does the Oracle version. If your site wants to run a more updated version of Oracle Java 8 then you will need to purchase a license from Oracle. In simple terms, the licence change introduced by Oracle in 2019 will not affect the running of your Weave system today, and not in the future.

Application Servers

Cohga supports the application servers listed on the page [Supported Platforms](#), provided they are running on Windows, Linux, a UNIX-based operating system (such as NetBSD, FreeBSD, OpenBSD and Solaris). If you have no preference for a particular application server and wish to set up Weave for production purposes, we highly recommend installing our standalone version, which includes the Jetty application server.

Weave supports the application server versions listed below. We may ask you to migrate to one of the supported application servers before we can provide you with further support.


- **Jetty** --- 8.1+
- **Apache Tomcat** --- 7.0+

Antivirus Software Configuration

The presence of antivirus software on your operating system running Weave greatly decreases the performance of Weave. Antivirus software that intercepts access to the hard disk is particularly detrimental and may even cause errors in Weave.

You should configure your antivirus software to ignore the following directories:

- Weave [home directory](#)
- All database-related directories

 The above recommendation is particularly important if you are running Weave on Windows. No matter how fast your hardware is, antivirus software will almost always have a negative impact on the performance of Weave.

Server Hardware Requirements

Minimum Hardware Requirements

On small instances, server load is primarily driven by peak visitors.

- 2GHz+ CPU
- 2GB RAM
- 5GB hard drive space

On larger installs where the number of entities, datasources, datadefinitions and users are upwards of 15 active concurrent application users.

- Quad 2GHz+ CPU
- 4GB+ RAM
- 10GB hard drive space

The recommendations above are for Weave only, and do not take into account other services/software that may be running on the server.

Connected users in passive mode

When a user opens a Weave client, a connection (which may not be persistent) would be maintained between the end user and the Weave server. The user may leave the Weave session idle without sending any query (e.g. search for a feature, pan the map). This case is an example of a connected user who is in a passive state. In the Weave user logs the statistics may show that 200 unique users have visited the application however, only a small portion of that number will be actively using the Weave concurrently.

Active concurrent application users

As in the instance described above, it is known that there are also users who would not just be connected to Weave but would be actively using Weave's features like adding or creating a selection, panning or zooming the map, searching an a-spatial database for example. It is during this kind of user connection state that the actual payload carried by the connection gets affected. *The higher the rate of user activity the higher will be the transactional throughput.*

Professional Assistance


For large instances, we suggest contacting Cohga for advice on hardware sizing, testing and performance tuning.

Client Software Requirements

Weave supports Web Browsers such as Internet Explorer (Microsoft Windows), Mozilla Firefox, and Chrome.

Please also read the additional information regarding client software requirements for Weave that are tabled on the Supported Platforms page.

Unsupported Web Browsers



 Internet Explorer 6, Mozilla Firefox 2 and Safari 2 will no longer be supported in future versions of Weave.



















Supported Platforms

This page describes the supported platforms for Weave as at November 2019. Please review these before installing Weave.

Key:  = Supported  = Not Supported

Java Version	
Java ⁽⁸⁾	 7 or 8 for Weave 2.5  8 for Weave 2.6

Operating Systems	
Microsoft Windows ⁽¹⁾	 32 and 64 for Weave 2.5  64 only for Weave 2.6
Linux ⁽¹⁾	 32 and 64 for Weave 2.5  64 only for Weave 2.6
MapEngines	
ArcIMS	 4.0, 9.0, 9.1, 9.2, 9.3, 10.0
ArcGIS Server	 10.1 or newer
WMS	 1.1.0, 1.1.1, 1.3.0 ⁽³⁾
SpatialEngines	
ArcSDE ⁽²⁾	 9.2, 9.3, 10.0, 10.1, 10.2
PostGIS	 1.4 or newer
Oracle Spatial	 10, 11, 12
SQLServer Spatial	 2008, 2012, 2014, 2016, 2017, 2019
Databases⁽⁴⁾	
PostgreSQL	 8.1 or newer
MySQL	 5.0.28 or newer
Oracle	 10, 11, 12
Microsoft SQL Server	 2005, 2008, 2012, 2014, 2016
DB2	 8.2, 9.7
Web Browsers	
Any modern browser ⁽⁵⁾⁽⁶⁾⁽⁷⁾	

1. Weave is a pure Java application and should run on this platform provided all other JDK requirements are satisfied.
2. Direct connect is not guaranteed to be supported for all versions of ArcSDE.
3. Depending upon the projection used.
4. Pretty much any database that provides a JDBC driver.
5. Internet Explorer must be at least version 8.
6. Except for Chrome 46.0.2490.86 on Windows.
7. Third-party application integration is only supported by Internet Explorer.
8. For more information in regard to Java Licencing go to [Server Software Requirements](#)

Example Size and Hardware Specifications

Below are some examples of Weave deployments that are currently being used

Num Users	Length of time in production	Application Server	Num CPUs/Cores	Physical Memory/RAM	Operating System
Local Government					
200 - 250	More than 3 Years	Jetty	2	8G	Windows 2003
Emergency Services					
200 - 300	More than 4 Years	Jetty	8	12G	Windows 2008

Installation

Installation

The Weave server uses [OSGi](#) (Open Services Gateway Initiative) to provide the internal infrastructure. The specific OSGi framework being used with the default Weave installation is [Eclipse Equinox](#).

OSGi provides a component integration platform that allows the Weave server to be delivered and updated in a modular format. In simple terms the installation of Weave is the installation of the underlying OSGi platform, followed by the installation of the Weave related bundles into this platform. In most cases these two steps are covered by copying a pre-prepared OSGi/Weave directory structure to the server.

While Equinox does provide a web server bundle to allow OSGi services to be made available over the web, Cohga has integrated the OSGi/Weave platform with a separate web server to provide additional functionality that is not currently supported by the Equinox HTTP web server bundle.

The default HTTP web server installed with Weave is Jetty but the server will also easily integrate with other (servlet supporting) web servers.

To integrate with an external web server, Weave provides a `weave.war` file that should be deployed to the nominated web server using whatever tools are provided by that web server. The default Weave directory installation structure already has a deployed `weave.war` file so in most cases this step will be unnecessary.

The `weave.war` file is a small web application that starts the OSGi platform and provides a link between the web server and the OSGi platform.

The default installation structure for Weave, which includes the OSGi/Weave platform and the Jetty web server, are setup so that the `weave.war` file can locate the platform directory when started. If a separate web server is used or a different directory structure is required, then it is possible to supply an `init` parameter to the `weave.war` file to tell it the location of the platform directory. Alternatively the entire Weave platform directory can be embedded within the `weave.war` file.

Starting the installer

The installer requires a Java runtime to be available to start the installer. If you're running the installer from the ISO image there is already a Java runtime available and will be used if the `setup.cmd` or `setup.sh` file is used to start the installer. If you're not using the .ISO image and are running the `weave-installer-x.y.z.jar` file directly then you will need to have a Java runtime already installed on the target server.

When Weave is installed it will install a Java runtime that is used to run Weave, unless you disable the option and point the installer to an existing Java runtime.

The installer determines if the server you're installing on is a 32-bit or 64-bit architecture and provides the appropriate options as to what Java runtime to install. It will only provide the 32-bit option if you're installing on a 32-bit machine, but will provide both 32-bit and 64-bit options if installing on a 64-bit server.

i If you're installing Weave onto a virtual machine the installer may think that you're running on a 32-bit server, when in fact you're using a 64-bit architecture, which will remove the option to install the 64-bit Java runtime.

To overcome this restart the installer with the system property `is64=true`, e.g.

```
java -Dis64=true -jar weave-installer-x.y.z.jar
```

Basic Directory Structure

Below is a cut-down directory structure showing the main directories and files for a default Weave installation

```

• weave
  • jre
  • jetty
  • logs
  • platform
    • configuration
    • features
    • plugins
    • workspace
      • config.xml
      • jdbc
  • webapps
    • weave.war
  • service
    • bin
    • bat
    • conf
      • wrapper.conf
  • startup.sh
  • startup.cmd
  • shutdown.sh
  • shutdown.cmd

```

`startup.sh`, `startup.cmd`, `shutdown.sh` and `shutdown.cmd` are Linux and Windows files for starting and stopping the Weave server from the command line. There are additional files in the `service` directory that will install Weave as a Windows service or Linux daemon. The `service\bin` directory is for Linux and the `service\bat` directory is for Windows. The `service\conf\wrapper.conf` file will configure the `service/daemon`.

The `weave.war` file is the web application described earlier, and `config.xml` is the main Weave configuration file.

The `workspace` directory also contains other resources that are related to the Weave configuration but generally do not need to be edited (e.g. the logging configuration file). Note that the `workspace` directory is also the default location used for report designs.

The `jdbc` directory contains Java JDBC drivers that Weave will use to connect to any databases. These drivers are not supplied with Weave and must be installed separately at installation time.

The `plugins` directory is where the code for the server is located. This directory contains all of the bundles that the server uses to provide the Weave functionality.

The `features` directory is a type of metadata directory about the bundles that are installed in the `plugins` directory, and it provides support for updating a group of bundles ("feature"), as a single entity.

The `configuration` directory, initially, contains a single file that OSGi uses to configure the startup of the OSGi platform.

To deploy Weave to an alternate Servlet Container such as Tomcat, deploy the `weave.war` file to that container. You will need to edit the `web.xml` file under the `BridgeServlet` to tell Weave where the platform directory is now located. An example of the xml code required is as follows:

```
<init-param>
    <param-name>platform</param-name>
    <param-value>/usr/me/weave/platform</param-value>
</init-param>
```

Note:

- current versions of Weave are compiled against Java 8 and will not work under Java 7 or earlier, or Java 9 and later.
- OSGi needs some environment variables set and `-Dosgi.clean=false` must be included in your JVM arguments
- if you need to connect to the OSGi console, then `-Dosgi.console=xxx` (where xxx is your desired port number (>1024)) must be included in your JVM arguments

Security

The security of the Weave web application is configured by a file named `security.xml`, which is located in the `workspace` directory.

The actual security infrastructure is provided by [Acegi Security](#), and the `security.xml` file is there to configure the Acegi components.

Setting up a second instance or Weave on the same server

In order to have two instances of Weave running on the same server the following procedures need to be undertaken.

1. Copy the Weave installation.

Copy the Weave installation directory to a new location. e.g. if I have a Weave installation in the following directory `c:\weave` and required a development instance, copy the `c:\weave` directory to `c:\weave_dev`.

Delete all files under the `c:\weave_dev\platform\configuration` directory, **EXCEPT** the `config.ini` file. The `config.ini` file **MUST NOT BE DELETED**.

2. Change the port numbers

Weave uses two port numbers by default unless changed in the Weave install wizard. Port 8080 for web access and 8070 is the jetty shutdown port.

To change the port number of the Jetty instance you can edit the `jetty.xml` file in the `weave_home` directory. In there you will see a section that looks like

```

<Call name="addConnector">
  <Arg>
    <New class="org.mortbay.jetty.nio.SelectChannelConnector">
      <Set name="host"><SystemProperty name="jetty.host"/></Set>
      <Set name="port"><SystemProperty name="jetty.port" default="
8080"/></Set>
      <Set name="maxIdleTime">30000</Set>
      <Set name="Acceptors">2</Set>
      <Set name="statsOn">>false</Set>
      <Set name="confidentialPort">${jetty.https.port}</Set>
      <Set name="lowResourcesConnections">5000</Set>
      <Set name="lowResourcesMaxIdleTime">5000</Set>
      <Set name="responseBufferSize">64000</Set>
    </New>
  </Arg>
</Call>

```

Change the instance of 8080 to a port that is currently not being used on the system.
e.g. the change would enable Weave to run on port 8081

```

<Call name="addConnector">
  <Arg>
    <New class="org.mortbay.jetty.nio.SelectChannelConnector">
      <Set name="host"><SystemProperty name="jetty.host"/></Set>
      <Set name="port"><SystemProperty name="jetty.port" default="
8081"/></Set>
      <Set name="maxIdleTime">30000</Set>
      <Set name="Acceptors">2</Set>
      <Set name="statsOn">>false</Set>
      <Set name="confidentialPort">${jetty.https.port}</Set>
      <Set name="lowResourcesConnections">5000</Set>
      <Set name="lowResourcesMaxIdleTime">5000</Set>
      <Set name="responseBufferSize">64000</Set>
    </New>
  </Arg>
</Call>

```

If you start Weave using the startup.cmd or startup.sh you will need to modify the shutdown port for jetty. If you start Weave as a Windows service there is no need to modify the shutdown port.

Within either the startup.cmd or startup.sh you will find the following. Change the 8070 to another free port on the server

```
JAVA_OPTS="$JAVA_OPTS -DSTOP.PORT=8070 -DSTOP.KEY=seekrit "
```

The shutdown.cmd or shutdown.sh will also need to reference this change.

```
JAVA_OPTS=" -DSTOP.PORT=8070 -DSTOP.KEY=seekrit "
```


If you start Weave as a Windows service then you should also edit the telnet port in `weave-service.conf`, if it's been set, so that it doesn't conflict with the copied instance. This is done by changing the 'osgi.console' value in the `weave-service.conf` file. Note this value may also be set in the `startup.cmd` file and should also be checked when copying an instance.

Additional Requirements

The OGR libraries, used to inter-operate with some spatial engines, requires the installation of the Microsoft Visual C++ 2013 Redistributable libraries.

They can be downloaded from [Microsoft from here](#).

Console and Automated Installation

Performing a console based installation

It's possible to install Weave from a console and not have to use the graphical UI. Note you will need to use Java 8 to run the installers.

To do this interactively you need to start the Weave installer with the `-console` command line option, e.g.

```
java -jar weave-installer.jar -console
```

This will walk you through an interactive installation process.

Automating the installation

To create an automated installation run the installer GUI and on the last screen save an installation script then edit that and use it on subsequent runs of the installer. It is not currently possible to create the installation script when performing a console install.



Using the Generate an automatic installation script button will ask you where you want to save the script, which is an xml file that can be edited to tweak the options used during subsequent installations.

You then start with weave installer with the location of the xml file you just saved on the command line option, e.g. assuming the above values saved in a file called `install.xml`

```
java -jar weave-installer.jar install.xml
```

This will perform a non-interactive console installation.

Upgrading

Upgrading from 2.4 to 2.5

✘ In Place Upgrade

It is not possible to directly update a Weave 2.4 instance to Weave 2.5. You must install Weave 2.5 to a new location, but you can copy your existing 2.4 configuration over to the new installation and you do not need to uninstall your previous Weave 2.4 instance.

✘ Installing on a virtual machine

Sometimes when installing Weave in a Virtual Machine the installer will incorrectly guess that you are installing on a 32-bit server and pre-selects the 32-bit Java Runtime rather than the 64-bit one.

If you are installing in a Virtual Machine when you get to step 4 in the installer ensure that you expand the Main components group and select the 64-bit Java Runtime.

✘ Network Ports

If you are installing Weave 2.5 on a server that already has a version of Weave installed, you need to ensure that either the ports used by the new instance are different from the existing instance, which you can change during the installation process, or that only one of the instances is running at a time.

To determine the ports that an existing Weave instance is using please see the page about [Network Ports](#).

⊕ Error messages early during Weave startup

You may notice a number of error messages that occur soon after Weave starts, similar to the following:

```
FrameworkEvent ERROR org.osgi.framework.BundleException: The bundle "org.eclipse.birt.report.data.oda.jdbc_4.3.1.v201308301349 [115]" could not be resolved. Reason: Missing Constraint: Require-Bundle: org.eclipse.core.runtime; bundle-version="[3.2.0,5.0.0]"
```

This is currently a known issue with the underlying OSGi platform and we're waiting for them to resolve the problem. Weave will still run correctly and the messages can be ignored.

Note: you can disable these warning for now by editing `...weave\platform\configuration\config.ini` and changing the line:

```
osgi.bundles=org.slf4j@1:start,\
to
osgi.bundles=org.slf4j@3:start,\
```

Installation

Starting the installer

To perform the initial installation you will require either the Weave installation ISO image or the Weave installer .jar file. Both can be downloaded from [Latest Downloads](#) if you have a current support subscription.

The ISO image will need to either be burnt to a CD (note that Weave 2.5 will fit on a CD, Weave 2.4 required a DVD) or mounted directly as an ISO image. To mount an ISO image in Windows see the documentation for [Mount-DiskImage](#).

The advantage of the ISO image is that it can be used to install Weave on a "clean" server, since it is self contained, whereas the Weave installer .jar file requires that the server have a version of Java already installed on the server, since the installer requires Java to run.

If you are using the ISO image then run the setup.cmd (on Windows) or setup.sh (on Linux) that is on the CD and the installer should start.

If you are using the .jar file then you will need to open a command prompt (sometimes double clicking on the .jar file may start the installer, depending upon how the server is setup) and start the installer manually.

As mentioned previously, the Weave installer .jar file requires an existing version of Java to be available on the server. If the server already contains a Weave instance then you already have Java available, and you just need to specify that Java be used when starting the installer.

For example, assuming that we have Weave 2.4 installed on Windows at `c:\weave-2.4\`, and the installer .jar file is currently located at `d:\temp\weave-installer-2.5.jar`, then you can initiate the Weave 2.5 installer with the following command:

```
c:\weave-2.4\jdk\bin\java -jar d:\temp\weave-installer-2.5.jar
```

Additionally, if you already know where you want to install the new Weave instance then you can also specify that on the command line, e.g.

```
c:\weave-2.4\jdk\bin\java -jar d:\temp\weave-installer-2.5.jar c:\weave-2.5
```

On Linux the process will be similar, e.g.

```
/opt/weave-2.4/jdk/bin/java -jar /tmp/weave-installer-2.5.jar /opt/weave-2.5
```

If you do not already have a Weave instance available on the server then you may need to install Java, which you can uninstall after you have finished the installation process.

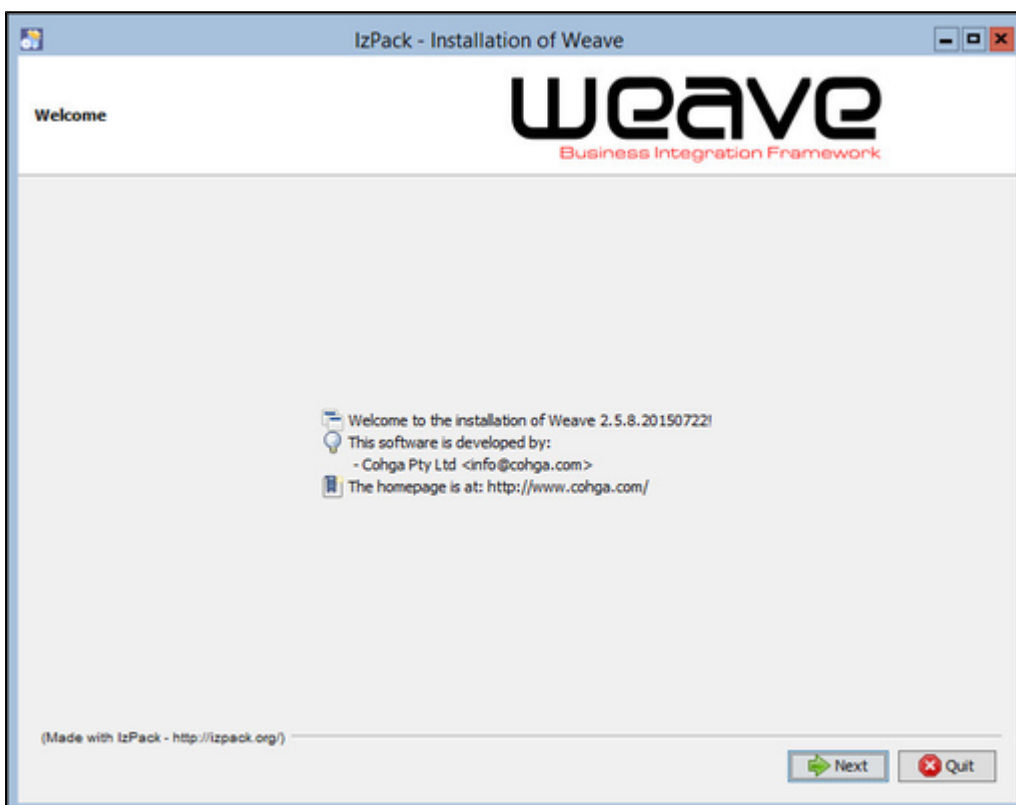
In this case the `java` command you will have to run will be different because the `java` executable will be installed in a different location, e.g.

```
"c:\program files\java\jre1.8.0_45\bin\java" -jar d:\temp\weave-installer-2.5.jar
```

The installation process

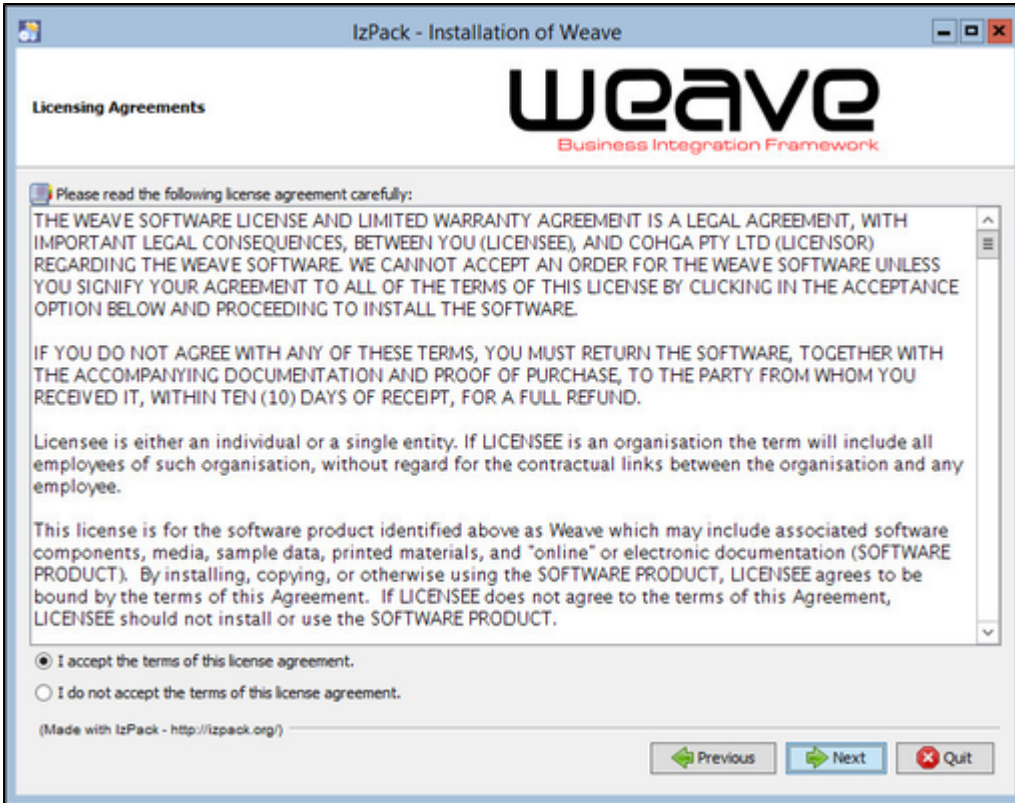
Step 1

Installing Weave 2.5 is similar to the process required for Weave 2.4, which starts with a welcome screen



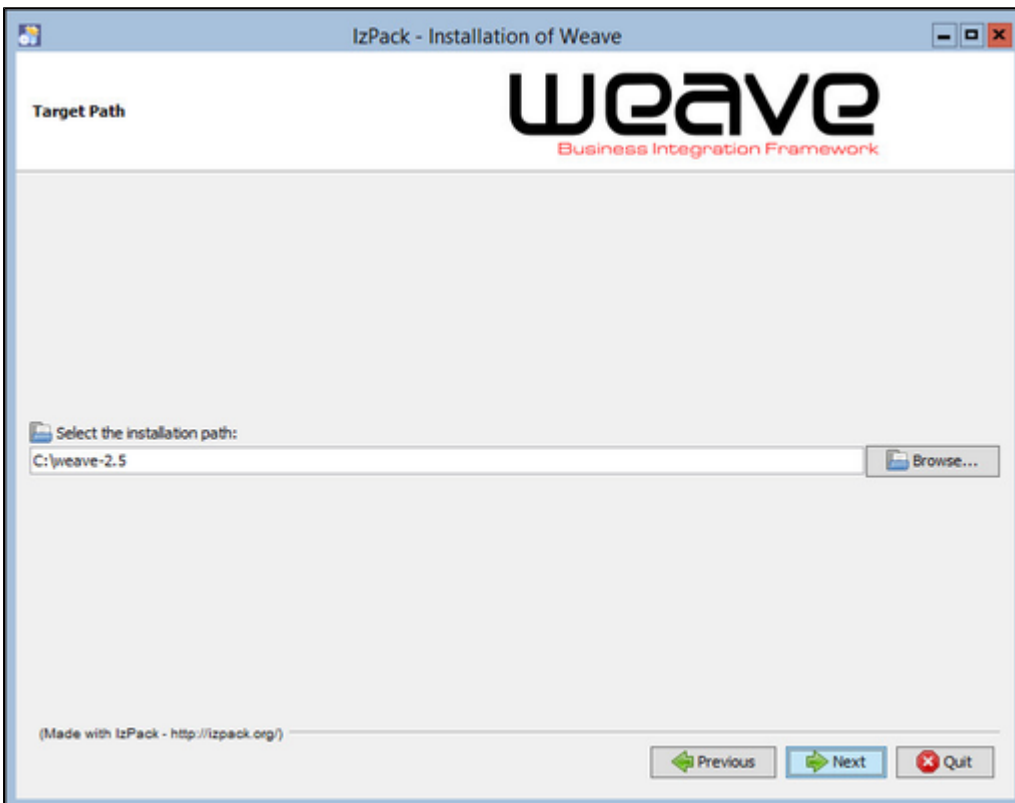
Step 2

This is followed by the Licence screen, where you must accept the licence agreement before you can proceed.



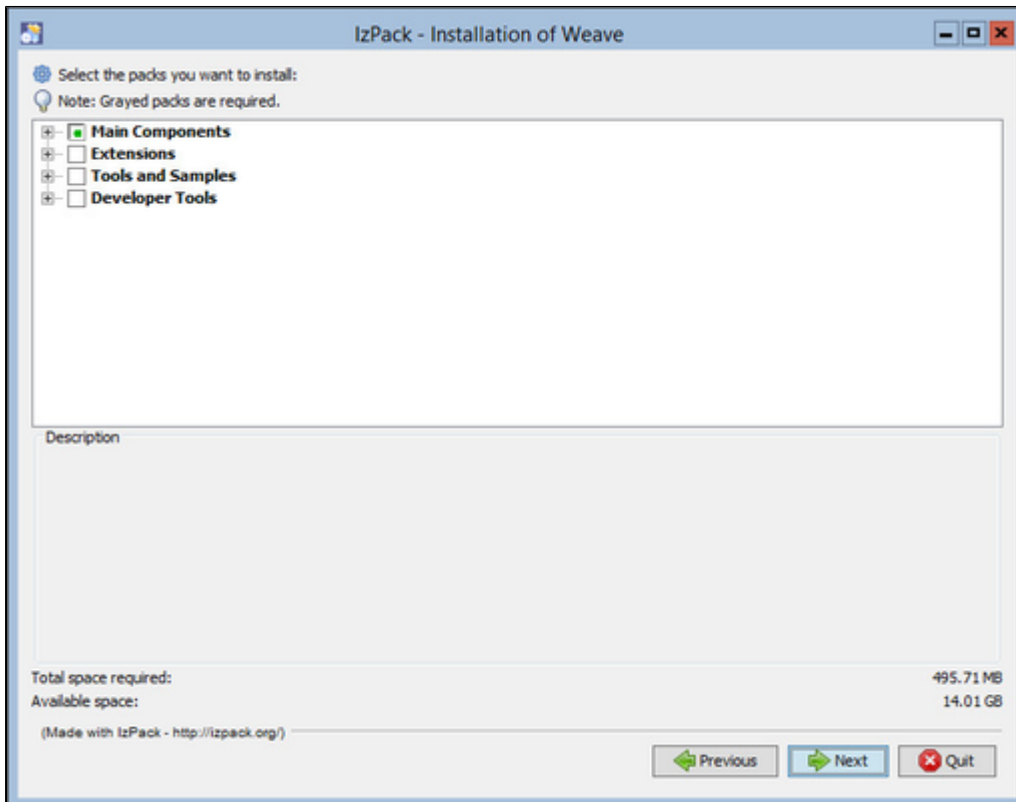
Step 3

Next you need to choose the location (Target Path) where you want to install the new instance of Weave. If you specified an install location on the command line when starting the installer, then the Target Path should already be showing on the Target Path menu.



Step 4

At this step you are presented with the available installation options.



The installation options are presented in a tree where the individual components are grouped based on their purpose.

The first group of options are the main components.

Virtual Machine Check

If you are installing Weave in a virtual machine it may be that the installer has incorrectly determined that you're running on a 32-bit operating system rather than a 64-bit one and has chosen the incorrect Java Runtime to install, or does not provide the option to install with a 64-bit Java Runtime at all.

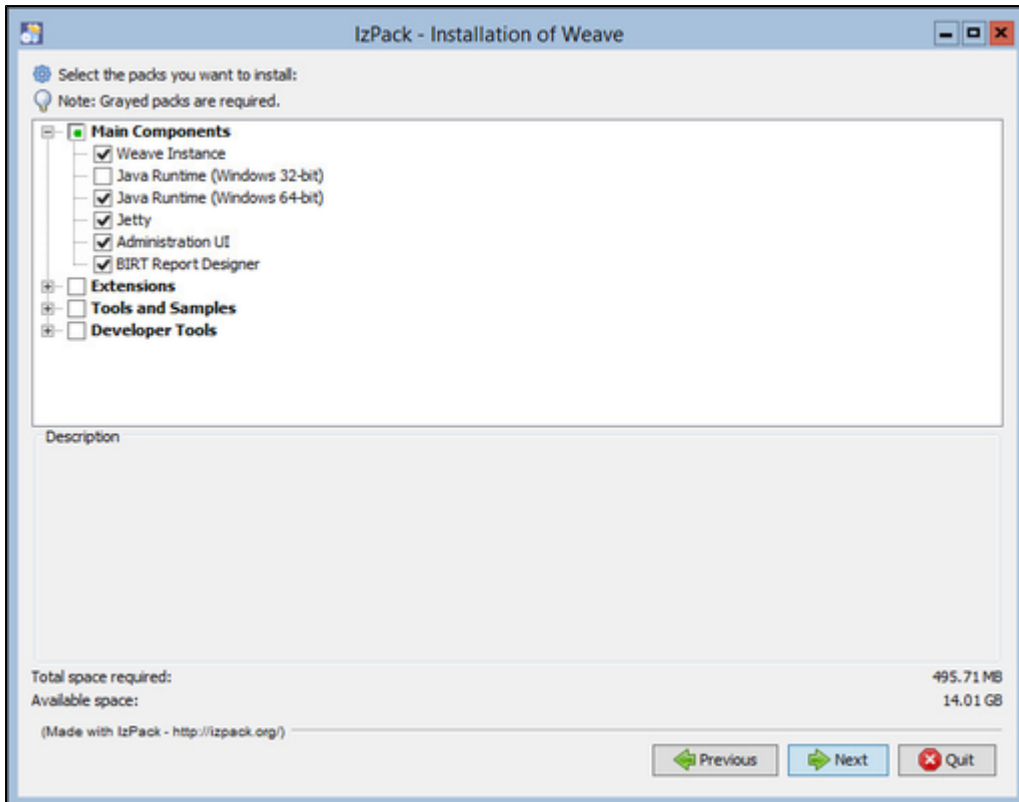
If it is possible you should change the Java Runtime in this group to switch to the 64-bit version.

If not, then you will need to restart the installer and force it to recognise the system as being 64-bit. To do this you should re-start the installer and add the following command line parameters:

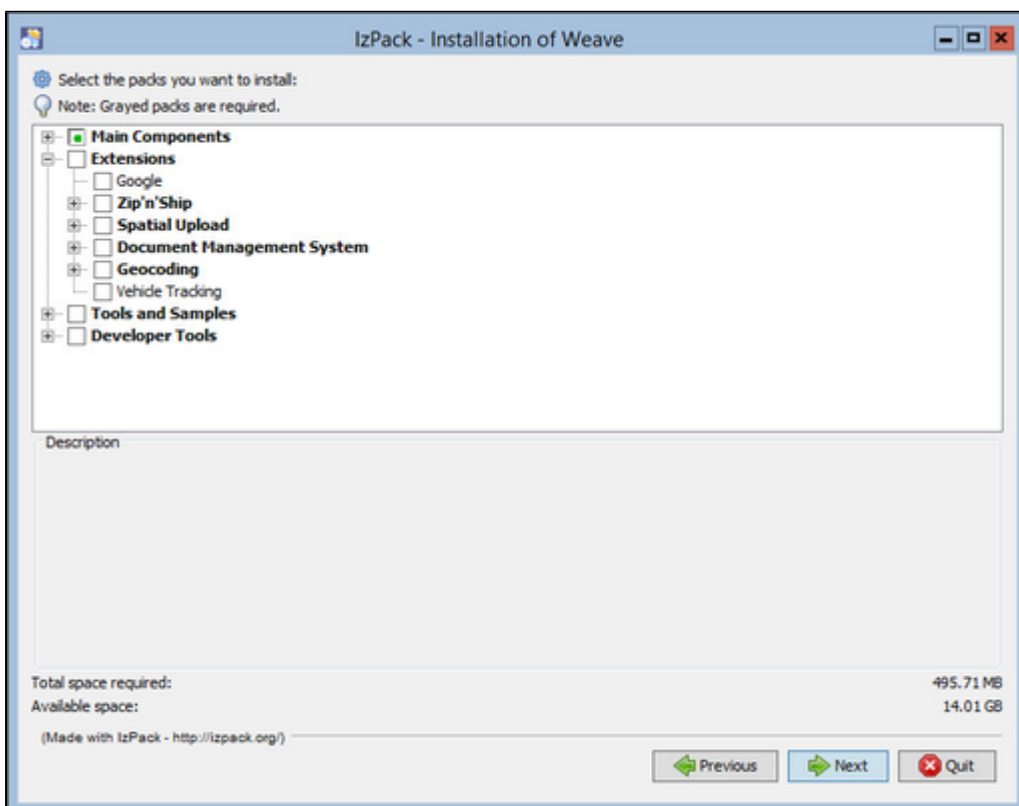
```
-Dis64=true
```

This will force the installer to assume that the server is 64-bit and not 32-bit. Similarly, if it is guessing 64-bit on a 32-bit server you can use `-Dis64=false` to force 32-bit as the default.

```
"c:\program files\java\jre1.8.0_45\bin\java" -Dis64=true -jar d:\temp\weave-installer-2.5.jar
```



The second group of options are the optional components that can be added to Weave. **Do not** just select all of them blindly. These are optional because they have a cost associated with installing them. If you do not need them then you should not install them. You **should**, since we are talking about upgrading an existing Weave instance here, select the same set of extensions that you had installed in the previous instance of Weave that you are upgrading.



The final two sections, 'Tools and Samples' and 'Developer Tools', do **not** need to be selected.

The Tools and Samples section contains a tool to convert an existing EView configuration to Weave, and some (old) example configuration files, neither of which are useful when upgrading an existing instance.

The Developer Tools section contains developer documentation and additional bundles that should only ever be installed when creating a Target Platform for creating Weave extensions in the Eclipse IDE. These should not be installed into a standalone instance of Weave and they are only required when setting up a development environment.

Step 5

The next step is to choose the port numbers that Weave will be listening on, normally these should be different from any other instance of Weave that is running on the same server.

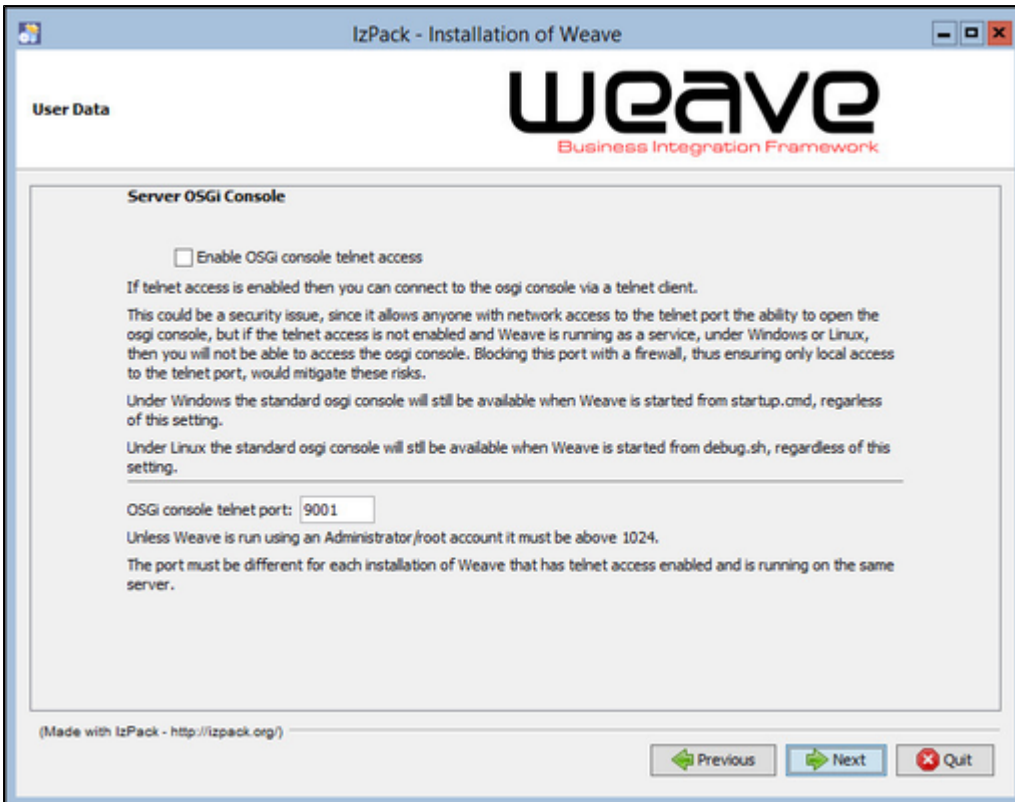
If you are performing this installation to upgrade an existing 2.4 instance, then you probably want to keep them the same port numbers as the existing instance so that you can just stop the old instance and start the new one and none of the users will have to be aware of any difference (for example, having to use a different URL to access Weave because the HTTP port has changed).

This can, however, cause problems if you want to run both instances at the same time. (e.g. if you are setting up the new Weave instance but still need to allow the users access to the old one). The obvious solution to this is that you probably should not be installing this new Weave instance directly in the same production server that your users are accessing and instead, should be using a staging/testing server for this process, and then moving the installation from the staging/testing server to the production server.

Assuming that you do not have this option, then you will need to specify a different set of ports at this step and once it is time to switch over, change the port numbers in the new instance to match the old one, stop the old instance, and start the new one (once you confirm that everything is working as it should).

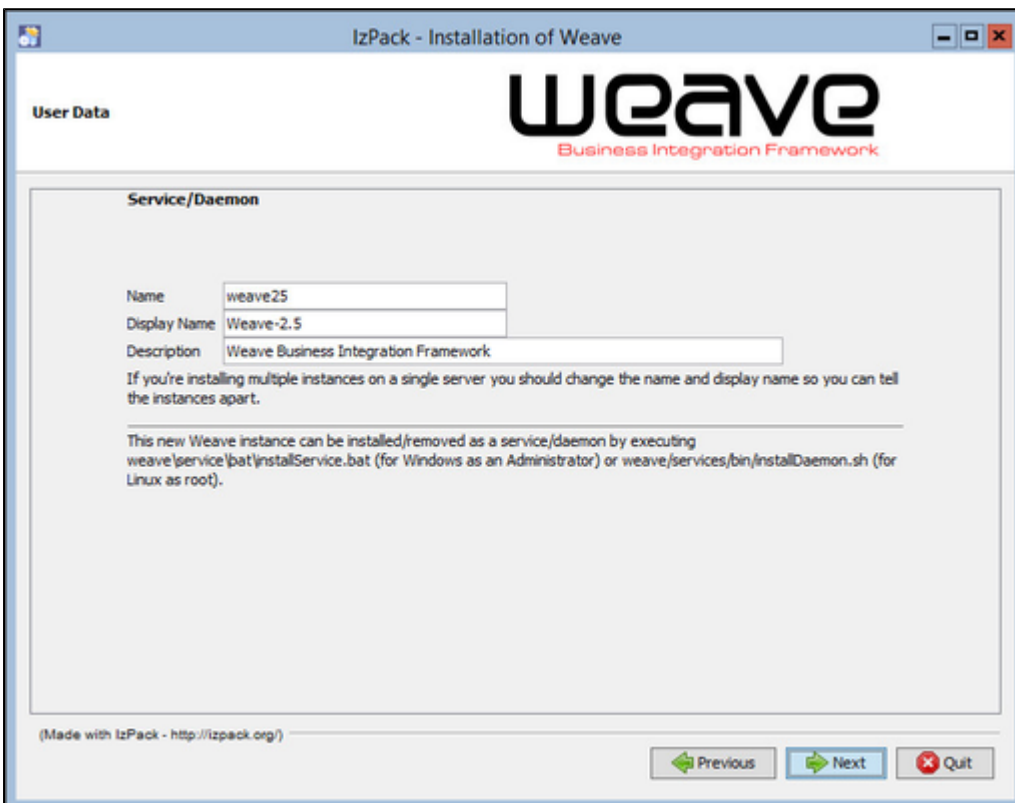
Step 6

The next step is to optionally enable the Telnet service and specify the port it will listen on. The same issues with this port number apply as those from Step 5.



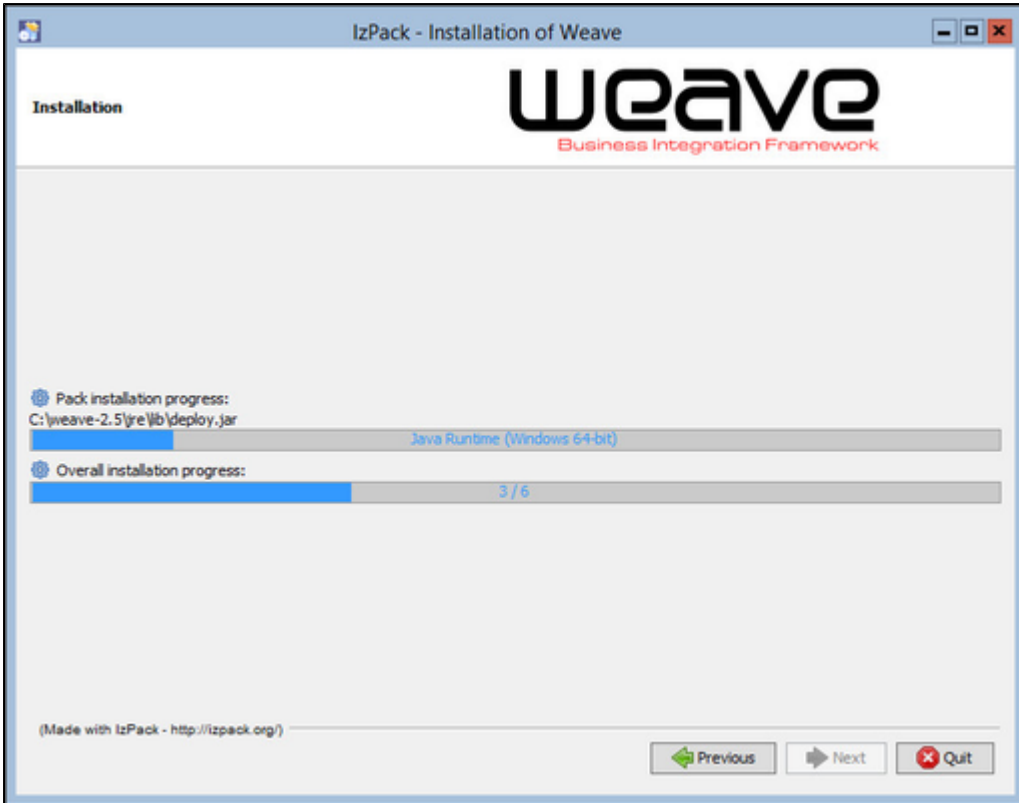
Step 7

This step configures the information required to install the new instance as a service/daemon. It does **not** install Weave as a service/daemon. That must be done as a separate step after the installation process is completed.



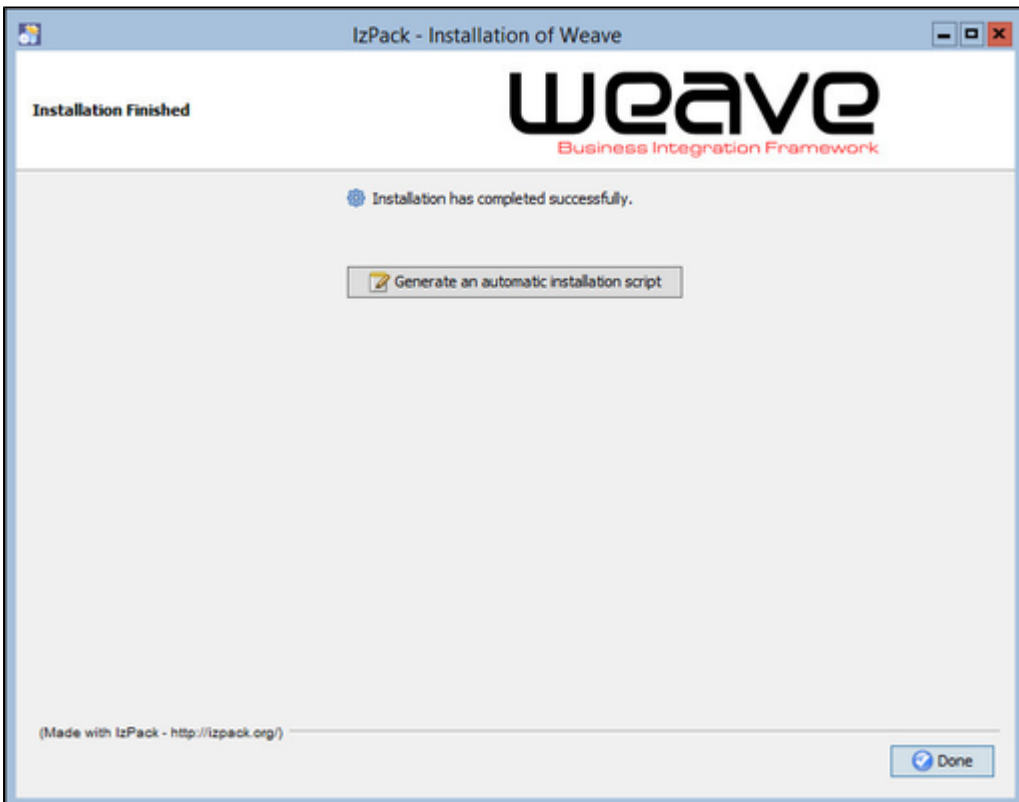
Step 8

The next page is shown when the installation process is running and the files are being copied to the new Weave installation directory.



Step 9

The final page is presented once the installation is completed and provides an option to generate an installation script. This script can be used with the installer at a later time to perform the same installation without having to go through the whole process of making the choices via the install GUI. In this way the installer can be run entirely without presenting the GUI and perform the same installation (with the option of installing to a different directory)



Interoperability with Third Party Applications

If you are integrating Weave with third party applications (e.g. Pathway, DataWorks, etc), then you will need to perform an extra installation step since these components are no longer provided as part of the core Weave installer. To do this, go to the download page and grab the

Interop installer download (it will be a .jar file you can run in the same manner outlined above for the main installer). You just need to provide the new Weave installation directory and which of the interop bundles you want installed as part of the installation process. The installer will install the additional bundles and update the installation to ensure that they are started.

Migrating your existing configuration

Once you have the new Weave instance installed you will need to move your configuration from the existing 2.4 instance to the new instance.

The primary files that you will need to migrate are `security.xml` and any files it includes (for example `users.properties`) and `config.xml` along with any files that it includes.

By convention some sites split their `config.xml` file into multiple files and stored most of the actual content under a `conf` directory. Generally it would be enough to copy `config.xml` and the `conf` directory from the previous instance to the new one.

If you have encrypted any passwords in your `config.xml` file using the OSGi consoles `encrypt` command then you will need to copy the `private.key` file from your old installation.

Once you have `security.xml` and `config.xml` (and their associated files) and maybe `private.key` copied over you should have enough to at least test that the new server will start.

To perform a test at this stage run the `startup.cmd` or `startup.sh` script to start the Weave instance. You should see a whole bunch of logging being output to the console (those keen eyed amongst you may notice a bunch of exceptions about bundle dependencies not being met logged early in the startup process, these can be ignored).

security.xml changes

There are a couple of places where the `security.xml` file has been updated from 2.4 so if you copy over your existing `security.xml` file you will need to apply these changes manually.

Near the top where the `filterChainProxy` is defined there are a couple of new values that need to be added to the filter list. By default there will be at least two entries in this list already, one for `/server/**` and one for `/**`. These entries describe the list of filters that must be applied to every client request that starts with this pattern, and we need to add a couple of new entries to this list.

The first entry to add is a completely new one

```
/server/ping=pingSessionContextIntegrationFilter
```

This should be added exactly as it's show above to the list. Additionally early releases of 2.5 did not include a similar entry for the `/server/heartbeat.do` pattern, you may need to manually add that entry too (see the more complete example below)

The second entry to add is to filter resources that start with `/services/**`, and for this filter we need to copy the existing `/server/**` filter and just change the pattern. So for example if your current `/server/**` filter is:

```
/server/**=httpSessionContextIntegrationFilter,
authenticationProcessingFilter,securityContextHolderAwareRequ...
```

you should copy and paste that line then change `/server/**` to `/services/**` in one of those lines, so you would end up with something like this:

```
/server/**=httpSessionContextIntegrationFilter,
authenticationProcessingFilter,securityContextHolderAwareRequ...
/services/**=httpSessionContextIntegrationFilter,
authenticationProcessingFilter,securityContextHolderAwareRe...
```

After those to changes the filter list should look something like:

```

/server/ping=pingSessionContextIntegrationFilter
/server/heartbeat.do=pingSessionContextIntegrationFilter
/server/**=httpSessionContextIntegrationFilter,
authenticationProcessingFilter,securityContextHolderAwareRequ...
/services/**=httpSessionContextIntegrationFilter,
authenticationProcessingFilter,securityContextHolderAwareRe...
/**=httpSessionContextIntegrationFilter,logoutFilter,
authenticationProcessingFilter,securityContextHolderAwa...

```

Note that the list of filters for `/server/**` and friends may be different for your `security.xml` file, for example yours may include `ntlmProcessingFilter` if you're using integrated authentication.

The ordering is important, in as far as ensuring that the more generic filter patterns are listed after the more specific ones, for example `/server/ping` being listed before `/server/**`, and `/**` being listed last.

After we've changed the filter list we need to add a new bean which provides the definition for the `pingSessionContextIntegrationFilter` we just added.

So after the `filterChainProxy` bean we just edited a new bean with the following definition:

```

<bean id="pingSessionContextIntegrationFilter" class="org.
acegisecurity.context.HttpSessionContextIntegrationFilter">
    <property name="allowSessionCreation" value="false"/>
    <property name="forceEagerSessionCreation" value="
false"/>
</bean>

```

That takes care of the first change that we needed to make, the second change is further into the `security.xml` file, and the thing we need to change is the `objectDefinitionSource` inside of the `filterInvocationInterceptor`.

The change is similar to that which we made to the `filterChainProxy` above, in that we're adding a couple of new filters that were not in the older file.

What we need to do is to ensure that three new URL's are always available regardless of the user being logged in or not, so the filter definitions we'll want to add are:

```

/login.*=IS_AUTHENTICATED_ANONYMOUSLY
/login/**=IS_AUTHENTICATED_ANONYMOUSLY
/report/remote/**=IS_AUTHENTICATED_ANONYMOUSLY

```

So after the changes you should have something like the following for the `objectDefinitionSource` within the `filterInvocationInterceptor`

```

        <property name="objectDefinitionSource">
            <!-- Make sure the login pages can always be
            accessed, unless you really want to disable forms login, and that
            BIRT can talk to the server -->
            <value>

CONVERT_URL_TO_LOWERCASE_BEFORE_COMPARISON
                                PATTERN_TYPE_APACHE_ANT
                                /login.*=IS_AUTHENTICATED_ANONYMOUSLY
                                /login/**=IS_AUTHENTICATED_ANONYMOUSLY
                                /report/remote
                                /**=IS_AUTHENTICATED_ANONYMOUSLY
                                /**=IS_AUTHENTICATED_ANONYMOUSLY
            </value>
        </property>

```

Note that you may have other entries in the list and `**` may have a different value, `IS_AUTHENTICATED_FULLY` for example, that doesn't matter, as long as you add the three new ones exactly as shown above, and add them before the `**` entry.

JDBC Drivers

You also need to copy any JDBC database drivers that Weave requires to connect to databases.

These drivers should be copied from the `platform\workspace\jdbc` directory in the old installation to the same directory in the new one.

One driver not to copy is the `org.hsqldb.hsqldb` driver, this is used internally by Weave and has been updated to a newer version and should already exist in the `platform\workspace\jdbc` directory.

Also, if you installed the BIRT report designer and you will be connecting to databases then you also need to copy your jdbc drivers into `birt\plugins\org.eclipse.birt.report.data.oda.jdbc_4.3.1.v201308301349\drivers`. This does not apply if you're just connecting to Weave to retrieve data, only if you are connecting to another database within any of your reports.

Testing

Once the server has finished starting up you should be able to open a browser and point it to the new instance and everything will appear as it did in the Weave 2.4 instance.

Yea right, if that happens I want to know, because there is probably going to be issues.

You will have bundles installed in your 2.4 instance that are not a core part of Weave that weren't installed as part of the installation, you'll have other custom files that were part of the old instance that have not been copied overs.

I'll attempt to update this document as sites come across these issues to provides clarity for other site, but for now it's probably a case of contacting support and outlining what issues you're having.

Additional configuration files

Aside from the security and config files there are a scattering of other files that you will probably want to look at when migrating from 2.4 to 2.5.

Encryption key

One important file that you will want to copy from your previous installation is the `private.key` file. This file contains the encryption key that Weave uses when it needs to internally encrypt content, and if you've been a good Weave administrator you've used the encryption facility provided by the OSGi consoles `encrypt` command to encrypt any passwords that you have in you `config.xml` file(s).

If you do not copy the `private.key` file from your previous instance then when Weave starts up it will generate a new encryption key, which means that when it tries to decrypt the passwords stored in the `config.xml` file(s) it will not generate the correct passwords.

Logging configuration

There are a number of changes between Weave 2.4 and 2.5 that have been made to the `logging.properties` file, primarily related to which packages are excluded from being able to generate log output.

If you copy your existing Weave 2.4 `logging.properties` from a 2.4 installation to a 2.4 installation you are going to have a large amount of extra mostly useless information output to the `weave.log` file (which does neither of us any good).

So rather than just copying your existing `logging.properties` file you should instead make any changes that you had made to the 2.4 version to the new 2.5 version (if any).

If in doubt I'd leave the file as it is rather than trying to guess, but if you insist on using a 2.4 `logging.properties` file then I suggest what you do is take the 2.4 version of the file and remove all of the lines that start with `log4j.logger.` then add in all of the lines that start with `log4j.logger.` from the 2.5 version, and use the combined content as your `logging.properties` file.

Post Installation

After you've completed the installation there are some additional steps that you may want to take.

Windows Service/Linux Daemon installation

You may want to install the instance as a service, this has changed from Weave 2.4.

Weave 2.5 uses a different wrapper to install Weave as a service than was used with Weave 2.4, and therefore the scripts and configuration used are different from earlier versions.

Everything related to installing Weave as a service is now contained within the `service` directory, this includes the configuration of the service, which is located at `service\conf` and the scripts for installing/uninstalling the service, which are located at `service/bin` (for Linux) or `service\bat` (for Windows).

The basic information required to configure the service were provided as part of the installation, but there are still numerous options that can be changed by editing the `service\conf\wrapper.conf` file, for example, this file can be edited to enable the Telnet service if it was not selected during installation, or it can be modified to change the account that is used to launch Weave.

More information about what can be configured for the service is available at [YAJSW Configuration Properties](#).

Finally, to install Weave as a service you need to run the `service\bat\installService.bat` script (on Windows) or `service/bin/installDaemon.sh` (on Linux).

Both of these scripts need to be run as an Administrator.

Weave Maximum Memory Limit

When Weave is run from the startup script the script contains a setting that specifies the maximum amount of memory the process will be allowed to use.

This was 1024M in Weave 2.4, and then changes to 2048M in Weave 2.5, but because it could cause issues when running on 32-bit server it has been reduced back to 1024M (from 2.5.9 onwards).

This may not be enough, and probably isn't but was chosen to ensure that the JVM had enough to at least start.

Depending upon how large your configuration is you may have to edit the startup script and increase the value.

Windows maximum memory limit set to 1024M in startup.cmd

```
SET JAVA_OPTS=-server -Xms512M -Xmx1024M -XX:PermSize=64M -XX:
MaxPermSize=192M
```

Linux maximum memory limit set to 1024M in startup.sh

```
JAVA_OPTS="-server -Xms512M -Xmx1024M -XX:PermSize=64M -XX:
MaxPermSize=192M"
```

When Weave is running as a service the maximum amount of memory it's allowed to use is not an absolute value (by default) but is instead set as a percentage of the total available memory.

This is specified in the `service\conf\wrapper.conf` file by the `wrapper.java.maxmemory.relative` value, which is set to 80 (percent) by default.

This value can be adjusted up or down or it can be commented out and `wrapper.java.maxmemory` used to set a fixed maximum (like the memory limit in the startup script).

Network Ports

Determining currently used network ports

A default installation of Weave uses up to three different network ports, one for the client to connect to, one for the shutdown process to connect to to tell the Weave instance to stop when Weave is started from the startup script, and one to provide telnet access to the running instance so that you can issue OSGi commands directly to the running Weave instance.

These ports must be different for each instance of Weave that is running on the same server at the same time or the server may not start. Depending upon if the instance is being started as a service/daemon or from the startup script the current values for these port numbers can be found in different locations.

Network ports for Weave 2.4 and 2.5

HTTP Port

The HTTP port that the client connects to is generally set in the `jetty.xml` file. One of the things that file configures is the connector for HTTP requests that will have a line like:

```
<Set name="port"><SystemProperty name="jetty.port"
default="8080" /></Set>
```

Where it's setting the port for the connector based in the system property names `jetty.port`, and if that value is not set then it will use 8080. 99% of the time there will not be a system property named `jetty.port` set, so 8080 (or whatever value you have) will be used. I'm going to assume for the moment that if you do have `jetty.port` set then you have enough knowledge to figure out what port numbers are being used and don't need my help in this section.

Shutdown Port

If Weave is started from the startup script, as opposed to being started as a service, then there is a setting in the `startup.sh/startup.cmd` and `shutdown.sh/shutdown.cmd` files which tells Weave the network port it should listen on when starting, and talk to when being ask to shut down. This is the stop port and will be specified as a system property in both the startup and shutdown files like this:

```
-DSTOP.PORT=8070
```

and should be kept as the same value in both the startup and shutdown scripts.

If Weave is running as a service on Windows then the service wrapper will take care of starting/stopping the the instance so you do not need to worry about the stop port. If Weave is running as a daemon on Linux then this will generally also be the case.

Telnet Port

If it is enabled then Weave will also listen on a network port using the Telnet protocol which allows you to communicate with a running Weave instance, which is particularly useful if Weave is running as a service/daemon.

If the Telnet port is not set the Telnet service will not be enabled, and you would not have to worry about the port conflicting with other Weave instances running on the same server.

Telnet port when running as a service

To find the current Telnet port number when running as a service/daemon you should look at the `weave-service.conf` file. If the port is enabled then there will be a `wrapper.java.additional` property set like:

```
wrapper.java.additional.12=-Dosgi.console=9001
```

The number 12 in this example may be different in your file, but if it is set then it indicates that the Telnet port is available and what port number it's listening on.

That the line may exist in that file but be commented out, if it's preceded by a # character, in which case the Telnet service is not enabled.
Telnet port when running from the startup script

The Telnet port can also be enabled in the startup scripts, but is less common. In this case the startup script will also set the `osgi.console` system property to the Telnet port number that it wants to listen on.

Telnet port set in Windows startup script

```
SET JAVA_OPTS=%JAVA_OPTS% -Dosgi.clean=false -Dosgi.console=9001
```

Telnet port set in Linux startup script

```
JAVA_OPTS="$JAVA_OPTS -Dosgi.clean=false -Dosgi.console=9001"
```

The startup script may have the `osgi.console` system property set, but not have it set to a port number, in this case it's enabling the OSGi console, but not starting the Telnet service to be listening on that port.

It may also not have the `osgi.console` value set at all, which also disables the Telnet service.

Upgrading from 2.5 to 2.6

Why Upgrade

Customers still on 2.5 will not have access to new features or updates to features that are in the 2.6 releases. Weave 2.5 will no longer be receiving any new functionality.

The functionality provided by Weave is supported by the underlying technology Weave is built upon, and provided as part of the base install of Weave. But as with all technology, these things sometimes need to be updated and as part of the release cycle of Weave it is now time for another one of these technology updates.

At the moment the core Weave functionality provided by Weave 2.5 and Weave 2.6 are exactly the same, that is they both contain the same "Weave" functionality, even though the underlying technology is being updated, and both versions are in fact made up of exactly the same set of bundles/plugins. However, from now on, all new development will be based on the updated 2.6 platform. As time goes by, the functionality provided by the new platform will be used more and more and will not be available for Weave 2.5, it is for this reason that upgrading from 2.5 to 2.6 should be done at the earliest opportunity, while the two releases of Weave are still closely aligned.

In-Place Upgrade

New functionality added to Weave requires updates to underlying technologies that Weave is built upon. These updates are generally made as part of a point release, e.g. 2.4 to 2.5, 2.5 to 2.6, etc., and as such it is not possible to directly update a Weave 2.5 instance to Weave 2.6.

You must install Weave 2.6 to a new location, but you can copy your existing 2.5 configurations over to the new installation and you do not need to uninstall your previous Weave 2.5 instance.

Note: If you're using encrypted passwords in your existing Weave configuration, you will either need to re-encrypt them in the new installation or copy the `private.key` file from the existing workspace to the new one. See [this page](#) for more information.

Supported Architecture

Weave 2.6 is only supported on 64-bit operating systems.

Network Ports

If you are installing Weave 2.6 on a server that already has a version of Weave installed, you need to ensure that either the ports used by the new instance are different from the existing instance or that only one of the instances is running at a time.

The port that Weave listens on can be set during installation, see step 5 below, or can be changed by editing `... \weave\jetty_base\start.d\http.ini` and changing the value of `jetty.http.port`, which defaults to 8080, and/or `stop.port` which defaults to 8070.

To determine the ports that an existing Weave instance is using please see the page about [Network Ports](#).

Base Directory Change

The base directory (the directory that is taken to be the "current working directory" when Weave is running) has changed from being in the `jetty` sub-directory to being the root of the Weave installation.

On a clean installation this will not cause any problems because the installer is aware of this change, but if you copy configuration files from a previous instance of Weave, for example `logging.properties` or `wrapper.conf`, and they contain relative paths, then these will now point to the wrong location.

You should check any configuration files that are being copied from an older Weave instance and ensure relative paths are changed to reflect the new startup directory. For example in `logging.properties`, the `log4j.appender.file.File` setting would change from `../logs/weave.log` to `./logs/weave.log` (note the two full stops have been replaced with one).

If in doubt, compare the new file installed as part of the clean Weave 2.6 installation with the old one you wish to replace it with before overwriting the newer file.

Additionally, support dumps will now be placed in the root directory rather than under the `jetty` sub-directory.

✘ Custom Bundles

Prior to Weave 2.6 if you were provided with a custom bundle (a plugin that contains code or enhancements specifically for your installation) you would copy it to the `...weave\platform\plugins\` directory and maybe edit the `...weave\platform\configuration\config.ini` file to have the bundle recognised and started by Weave.

This is no longer the case. Instead custom bundles should be copied to the `...weave\platform\custom\plugins\` directory and `...weave\platform\configuration\config.ini` **should not** be changed.

✘ Cleaning the configuration directory

Previously if there were issues with starting Weave you may have been asked to "clean up the configuration directory", that is delete everything in the `...weave\platform\configuration\` directory except for the `config.ini` file.

You should not do that with Weave 2.6.

There is now an additional directory within the configuration directory that should not be deleted, the `felix.fileinstall` directory.

You should not have to clean out the configuration directory any more, but if you do then make sure not to delete either the `config.ini` file or the `felix.fileinstall` directory.

📌 Weave 2.6 only updates some of the underlying infrastructure of Weave, the actual Weave code is the same as Weave 2.5. The updates are for the:

- Java Virtual Machine, which is updated from 7 to 8,
- Jetty Web Application Server, which is updated from 8 to 9, and
- Java Service Wrapper, which is updated from 11 to 12.

📌 As part of the update of Jetty and the Java Service Wrapper a number of system properties that were previously set in `jetty.ini` and `jetty.xml`, `startup.cmd/startup.sh` or `wrapper.conf` are now set as part of Jetty at `jetty_base\start.d\`.

Note that if you're using a Web Application server other than the Jetty installed as part of Weave then you will also need to duplicate some of the settings within the other Web Application Server.

HTTP Settings

The settings related to the HTTP server, which ports to listen on, etc., were previously set in `jetty.ini` or `jetty.xml`. These values are now stored in `jetty_base\start.d\http.ini`.

Proxy Settings

The settings required for the Weave server to know which HTTP proxy to use to access the internet, if required, were previously set in `startup.cmd/startup.sh` and `service\conf\wrapper.conf` but are now stored at `jetty_base\start.d\proxy.ini`.

Weave Settings

There are a number of customisable settings available for Weave, which were previously set in `startup.cmd/startup.sh` and `service/conf/wrapper.conf` but are now stored at `jetty_base\start.d\weave.ini`.

If you have your own settings that you have added to `startup.cmd/startup.sh` and/or `service/conf/wrapper.conf` you should create your own `.ini` file, using something like "custom.ini" or "organisation name.ini" at that location and add your setting to that file.

Telnet Settings

~~The setting related to enabling telnet access to the Weave server OSGi console was previously set in `startup.cmd/startup.sh` and `service/conf/wrapper.conf` but is now stored at `jetty_base\start.d\telnet.ini`.~~ There appears to be an issue with starting the Telnet service when this setting is moved so it had been moved back until the issue is resolved.

Warning: The ini files referenced above must use ISO-8859-1 encoding, not UTF-8. Ensure that the text editor you're using to edit the files is using the correct encoding when saving the files or the file contents may not get processed.

Installation

Starting the installer

To perform the initial installation you will require the Weave installer .jar file. This can be downloaded from [Latest Downloads](#) if you have a current support subscription.

To use the .jar file you will need to open a command prompt (sometimes double-clicking on the .jar file may start the installer, depending upon how the server is setup) and start the installer manually.

The Weave installer .jar file requires an existing version of Java to be available on the server. If the server already contains a Weave instance then you already have Java available, and you just need to specify that Java is used when starting the installer.

For example, assuming that we have Weave 2.5 installed on Windows at `c:\weave-2.5\`, and the installer .jar file is currently located at `d:\temp\weave-installer-2.6.30.jar`, then you can initiate the Weave 2.6 installer with the following command:

```
c:\weave-2.5\jre\bin\java -jar d:\temp\weave-installer-2.6.0.jar
```

Additionally, if you already know where you want to install the new Weave instance then you can also specify that on the command line, e.g.

```
c:\weave-2.5\jre\bin\java -jar d:\temp\weave-installer-2.6.0.jar c:\weave-2.6
```

On Linux the process will be similar, e.g.

```
/opt/weave-2.5/jre/bin/java -jar /tmp/weave-installer-2.6.0.jar /opt/weave-2.6
```

If you do not already have a Weave instance available on the server then you may need to install Java, which you can uninstall after you have finished the installation process.

In this case the `java` command you will have to run will be different because the `java` executable will be installed in a different location, e.g.

```
"C:\program files\java\jre1.8.0_45\bin\java" -jar d:\temp\weave-installer-2.6.0.jar
```

There is also a ISO file installer available that contains a Java runtime that can be used if the server does not have Java already available or you do not want to install an additional Java runtime on the server to install Weave.

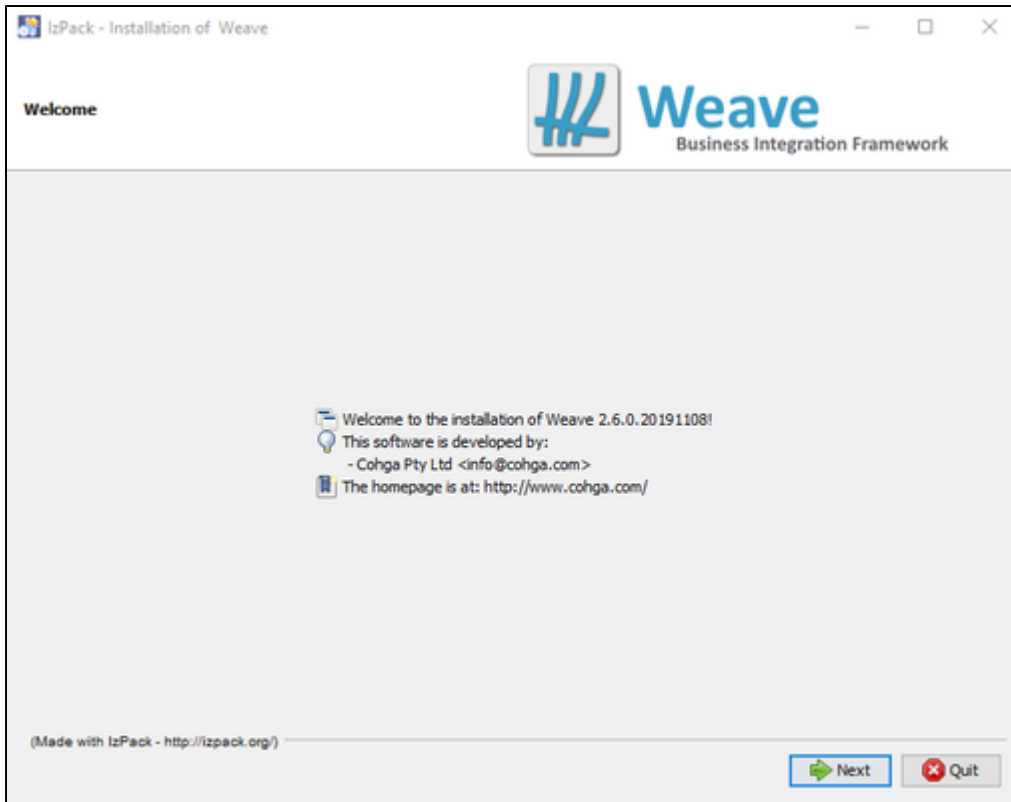
The installer can be mounted on the server and should appear as a separate drive, on Windows, or directory, on Linux, and contains a `setup.exe`, for Windows, and `setup.sh`, for Linux, that can be run to initiate the installation process.

Note that in Windows 8 and Windows Server 2012 and later it is possible to mount an ISO file without having to install a third party tool by selecting the .iso file in Windows Explorer and right clicking on it and selecting "Mount ISO", if this is not available, which can happen if another program has taken over support for opening ISO files, then you can select "Open with" and choose "Windows Explorer".

The installation process

Step 1

Installing Weave 2.6.0 is similar to the process required for a standard Weave upgrade, which starts with a Welcome screen.



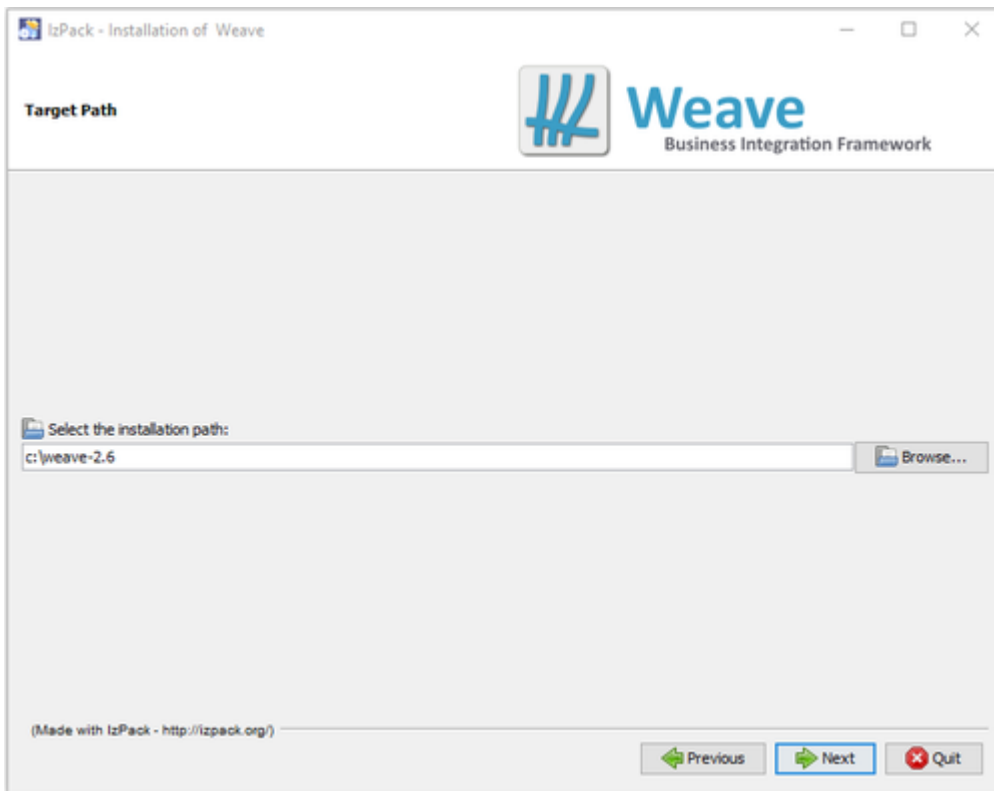
Step 2

This is followed by the Licence screen, where you must accept the licence agreement before you can proceed.



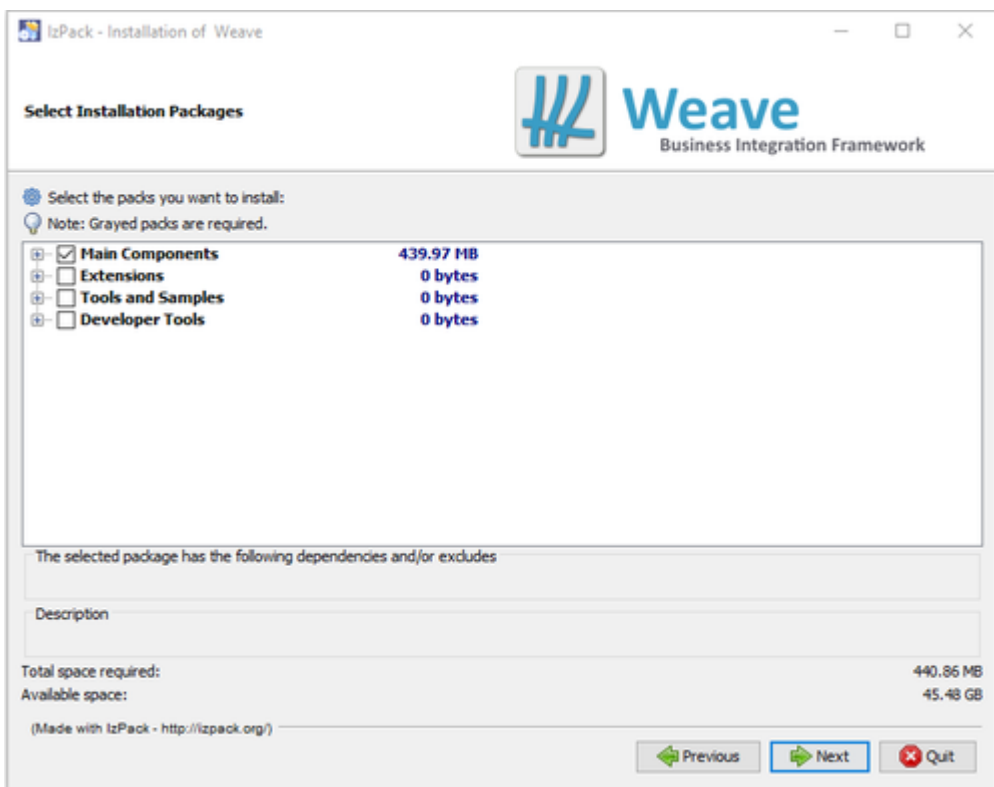
Step 3

Next you need to choose the location (Target Path) where you want to install the new instance of Weave. If you specified an install location on the command line when starting the installer, then the Target Path should already be showing on the Target Path menu.



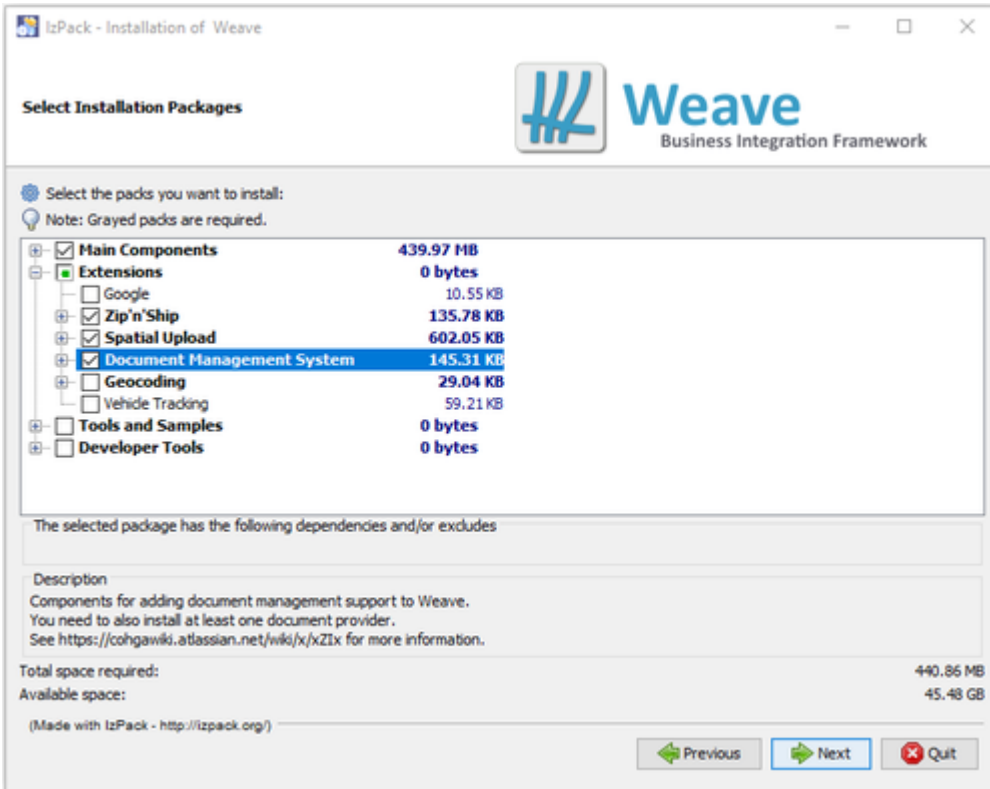
Step 4

At this step you are presented with the available installation options.



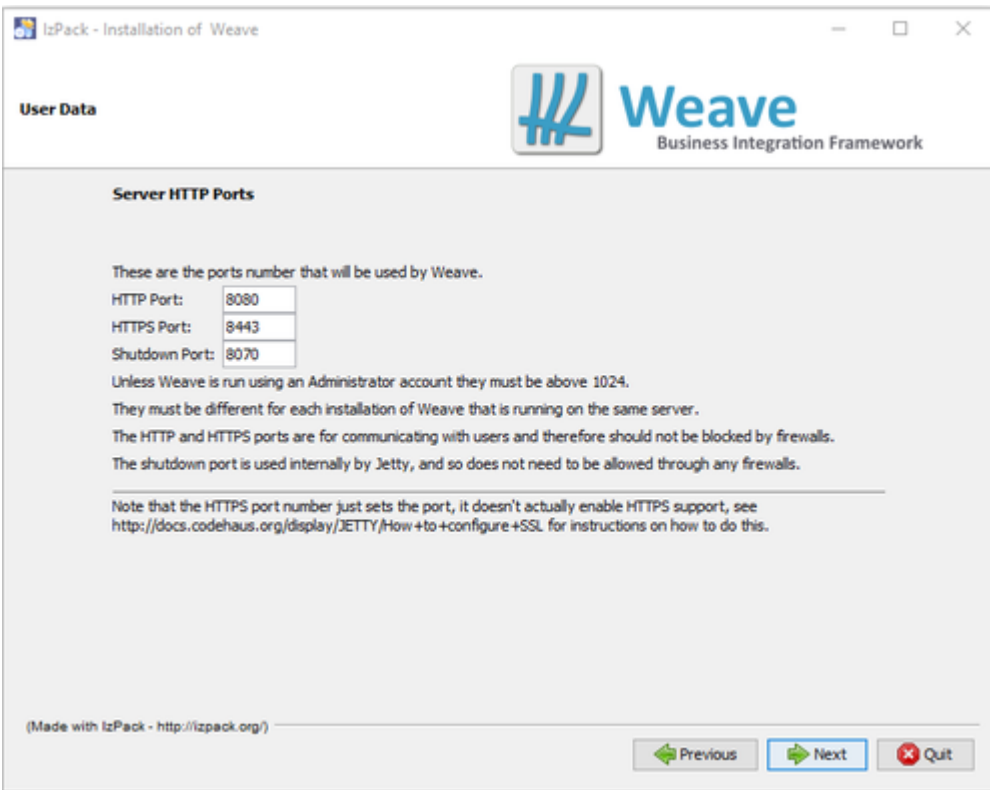
The installation options are presented in a tree where the individual components are grouped based on their purpose.

The first group of options are the main components. Since we are upgrading an existing Weave instance, select the same set of extensions that you had installed in the previous instance of Weave that you are upgrading.



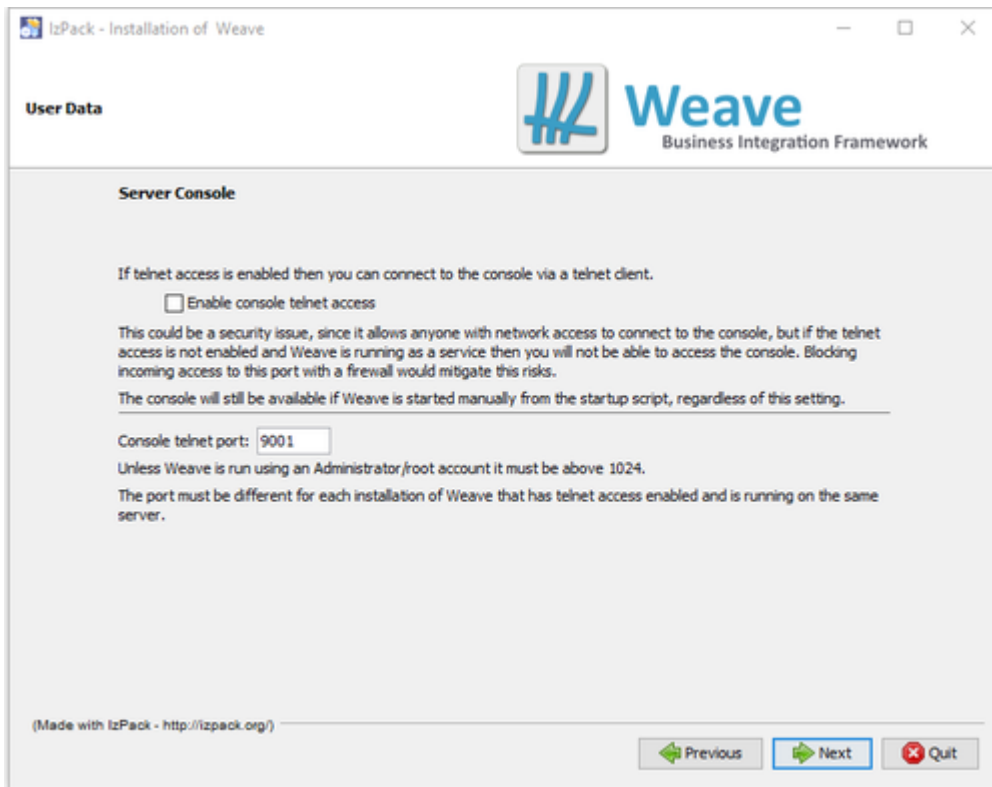
Step 5

The next step is to choose the port numbers that Weave will be listening on. If you choose the same as those used for your existing Weave instance you can only run one instance of Weave on that server at the one time.



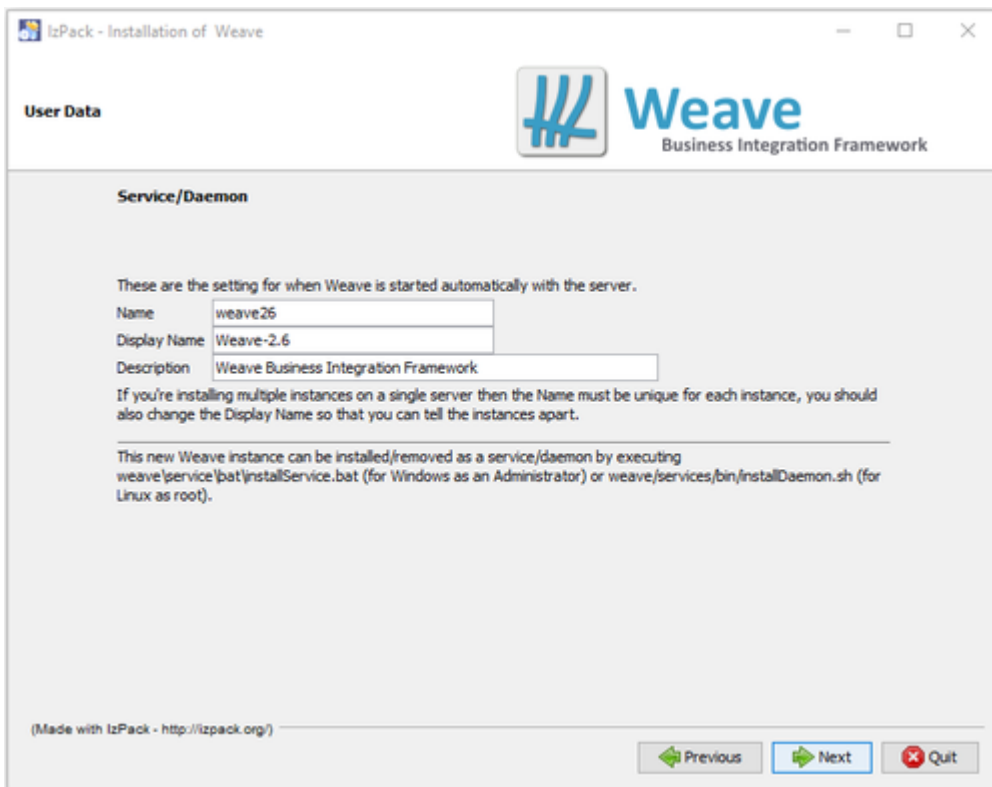
Step 6

The next step is to optionally enable the Telnet service and specify the port it will listen on. The same issues with this port number apply as those from Step 5.



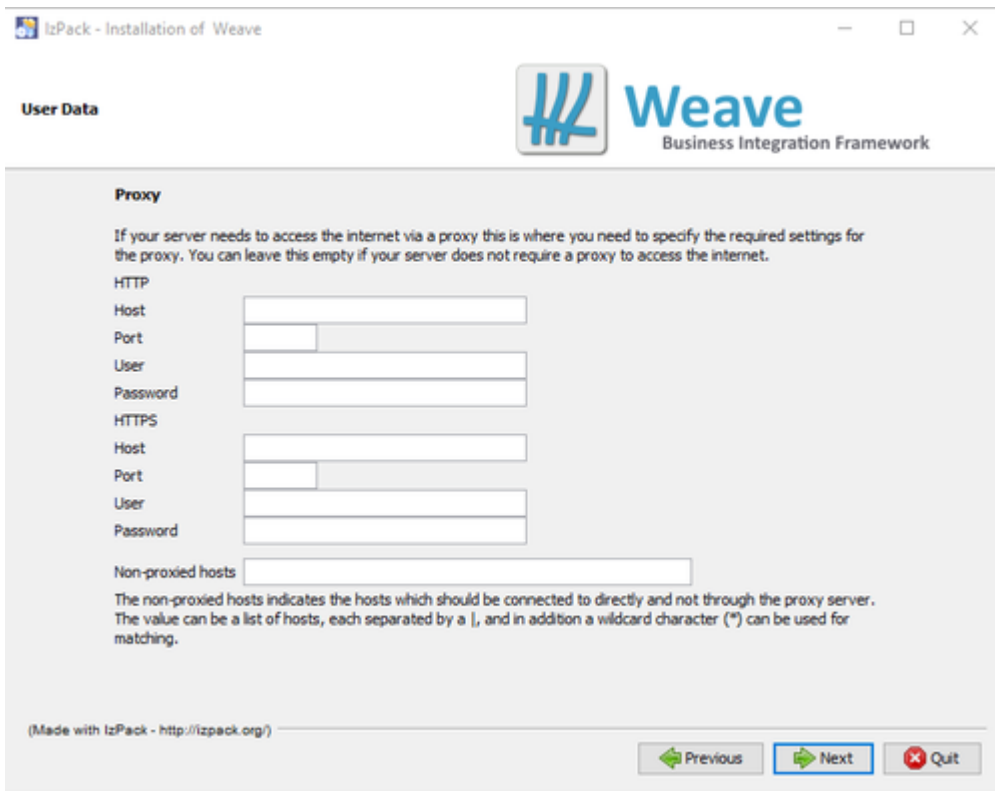
Step 7

This step configures the information required to install the new instance as a service/daemon. It does **not** install Weave as a service /daemon. That must be done as a separate step after the installation process is completed.



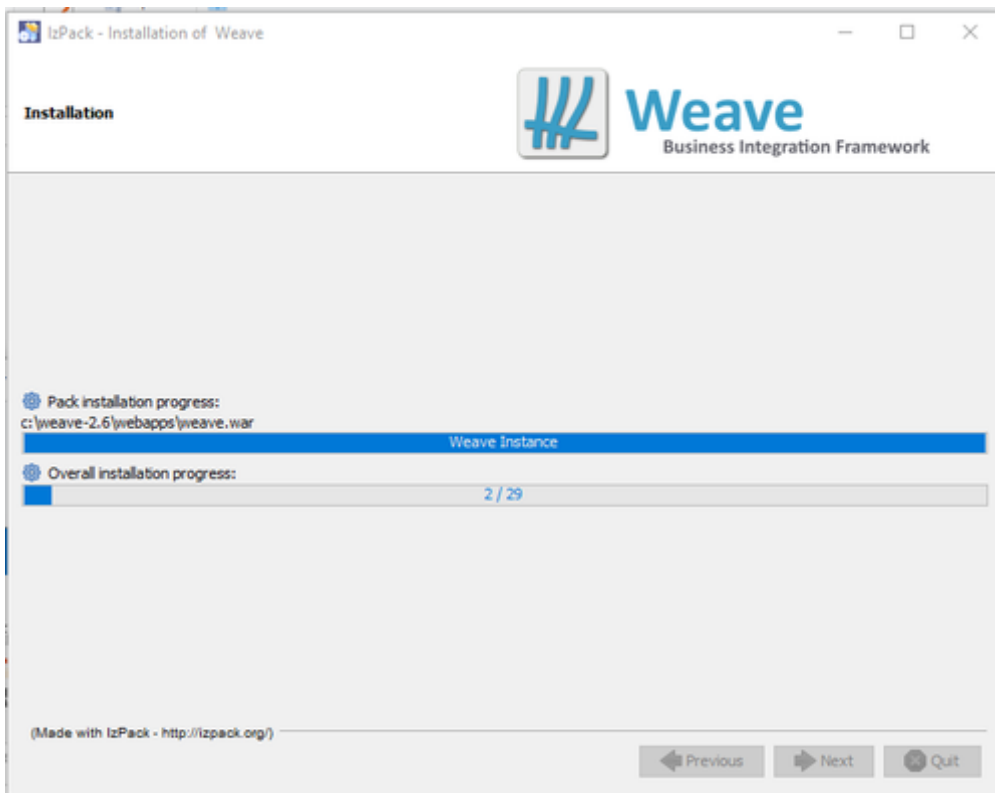
Step 8

If your server needs to access the internet via a proxy you need to fill out the required settings.



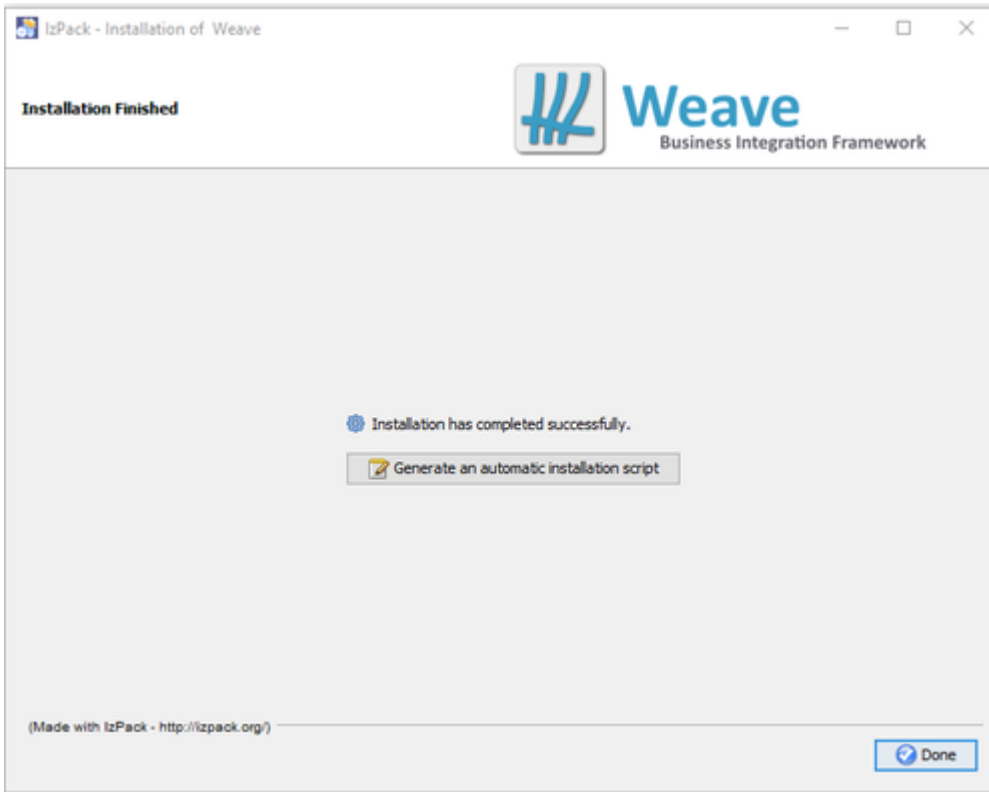
Step 9

The next page is shown when the installation process is running and the files are being copied to the new Weave installation directory.



Step 10

The final page is presented once the installation is completed and provides an option to generate an installation script. This script can be used with the installer at a later time to perform the same installation without having to go through the whole process of making the choices via the install GUI. In this way the installer can be run entirely without presenting the GUI and perform the same installation (with the option of installing to a different directory).



Latest Downloads

Go to the [What's New](#) page to see what's changed in these updates.

Current Release - 2.6

i In order to improve the download speed, and help automate the processing of new releases, information about the latest stable release and links to download the files have been moved to a [new page which you can find here](#).

Note that the 2.6 updater (or installer for that matter) can **not** be applied to an existing 2.5 installation, you need to install 2.6 in a separate directory, but there's no requirement to uninstall your existing 2.5 instance before (or after) doing so.

For more information on upgrading from 2.5 to 2.6 see [this page](#).

Once 2.6 is installed you can copy your existing 2.5 xml configuration files to the new 2.6 instance, but **you should take note of the changes in files installed by the installer compared to your files** (for example security.xml, logging.properties, export_html.css etc), as some of these files have changed since 2.5 and you may need to apply these changes to your 2.5 versions before they can be used in 2.6. Generally the files will work as is when copied from 2.5, but it may provide a more optimal experience if you check for differences.

The interoperability installer contains the bundles required to integrate with third party applications, e.g. Pathway, ProClaim, etc.

The installer and updater will accept a directory name as a command line parameter. For the installer the parameter will be used as the default for the target directory to install to, but more importantly, for the updater, if the directory points to an existing Weave installation it will automatically update it without requiring any user intervention.

Updating a Weave instance from the command line on Windows

```
java -jar weave-updater-2.6.1.jar c:\weave
```

Updating a Weave instance from the command line on Linux

```
java -jar weave-updater-2.6.1.jar /opt/weave
```

File	Description
ISO	This is a CD image that can be mounted or burned. It contains the main Installer and allows you to install Weave without already having Java installed.
Installer	This is the main Installer. If you already have Java installed, it can be used to install Weave.
Interop	This is a separate installer that is used to install third party application interoperability plugins. To add third party interoperability plugins, run the installer after the main installer.
Edit	This is a separate installer that is used to install the editing related plugins. To add the editing related plugins, run this installer after the main installer.
Updater	This will update a Weave installation to the latest release. Updates to plugins that are added from the Edit and Interop files are included, as well as the updates for the main Installer.
Hotfix	Hotfix is similar to the Updater, but applies to a specific long term support release and only contains bug fixes.
Designer	The BIRT report designer, which used to be included in the Installer but now has its own separate installer.

End of Support Announcements for Weave

Deprecated Web Browsers for Weave

This section announces the end of Cohga support for certain web browsers for Weave.

End of Life Announcement for Web Browser Support	Web Browsers	Support End Date
	Internet Explorer 6	13 July 2010

- **Internet Explorer 6 Notes:**

Cohga will support IE6 in Weave until the 13th of July 2010. This in line with Microsoft's [Support Lifecycle policy](#). Beyond this date, released versions of Weave will continue working with IE6 just as they did before, but we will not be in a position to fix problems in use cases that are affected by Internet Explorer 6.

You may be able to use Internet Explorer 6 for the most common use cases, but official support for this browser will end once you next upgrade.

Weave Hub - Third-party Application Integration

Weave Hub

Background

Integration between Weave and third-party applications previously relied upon [Weave Link](#) to provide a means of communication between the third party application and the browser running Weave. However, Weave Link was limited in that it required Internet Explorer to be used as the browser. To address this limitation and open up third-party integration in Weave to additional browsers Weave Hub has been developed. Weave Hub is an application that runs on each client PC and provides a communications channel between the browser and the other applications. Weave Hub by itself only supports communications between a browser running the Weave client and Weave Hub. To allow Weave Hub to communicate with other applications additional add-ins must also be installed.

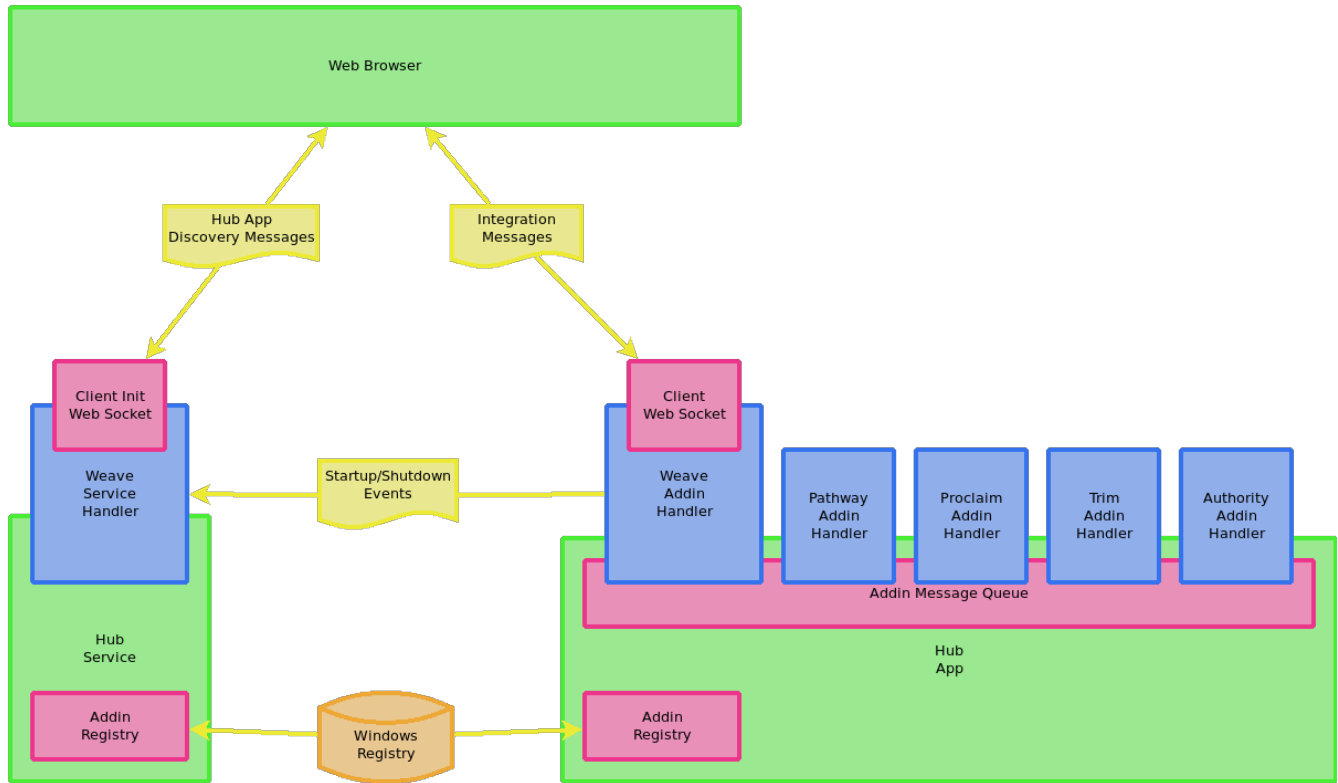
Basic Design

The basic design of Weave Hub is significantly different from that of Weave Link. Weave Link was a component that was installed locally on the client PC and third-party applications used it to directly communicate with Weave. The Weave Link component was used by the third-party applications and would communicate directly with the browser (which is why it was limited to Internet Explorer). Weave Link would take

care of locating an existing browser and starting a new browser session if one could not be located and then access the JavaScript code embedded within the Weave application. Weave Hub, however, is an application that runs locally on the client PC and listens for messages and handles the routing of those messages between the appropriate applications. Out of the box, Weave Hub knows how to handle messages sent to and from Weave and, by installing additional add-ins, messages for other applications can also be handled.

Note that currently Weave Hub is supported on 32 and 64 bit Windows but currently only 32 or 64-bit Weave Hub add-ins can be used at one time, and it requires at least version 4.6.1 of the .Net Framework to be installed (which should be installed by default on Windows 10 and installed via a Windows Update for Windows 7 since around January 2016). The choice between the 32 or 64-bit installation depends upon the third-party applications that you use, not the operating system or browser. If you run a 32-bit version of a third-party application, even if it's on a 64-bit operating system with a 64-bit browser, then you'll need to use the 32-bit version of Weave Hub and the associated plugins. Note this can be an issue if you run a mix of 32 and 64-bit third-party desktop applications.

The diagram below shows parts required for Weave Hub and how the communication between them occurs.



Weave Hub Installation

Client PC Components

Installation of Weave Hub is performed via a .msi file which only provides options to change the default installation location and whether or not to launch Weave Hub after installation is completed. The installer will, by default, install the core Weave Hub components at C:\Program Files\Cohga\WeaveHub\, and will create a shortcut in the Startup folder to the Weave Hub application so that it's launched each time the user logs in along with a Windows service. Additionally, it creates some registry entries, at HKLM\SOFTWARE\Cohga\WeaveHub. The installation can be performed silently but in this case, it will not be started automatically. **The WeaveHub.msi installer must be run as an administrator.**

To obtain the required client installer .msi files and install any server bundle updates, you need to download the Interop installer package for your version of Weave from <http://downloads.cohga.com/weave/>. Once you've downloaded the file you should run the Interop package on the Weave server in the same way you would if you were installing or upgrading Weave.

i If you're not running the latest version of Weave and need to download the Interop installer for a previous release, you can copy and paste the link to the Interop installer from the download page and change the version number embedded in the link to match the version of Weave you're using.

The Interop installer requires that you provide it with the path where your Weave server instance is installed and that you select the third-party applications you wish to integrate with. Ensure that the server is not running when you perform the installation. Once the Interop installation is complete, any required Weave server bundles will have been installed in the ...weave\platform\plugins directory, and the required client installer .msi files copied into the ...weave\interop directory. You can now copy the .msi files from those directories and apply them to the client PC using your internal software distribution processes.

Weave Server Components

To provide support to the rest of the Weave client, an updated version of the `com.cohga.client.interop` bundle, at least version 3.0.0, must be installed in the Weave server. This updated bundle provides code that the third-party components within the client will use to communicate with the Weave Hub. Additionally, the Weave Hub integration must be enabled via a plugin to the client configuration. This can be done by adding the `weave.hub` plugin to the client as shown below.

Adding Weave Hub plugin to client configuration

```
<client:config id="main">
  <title>Weave</title>
  <description>Main Weave Client</description>

  <plugin id="weave.hub"/> <!-- enable the weave.hub plugin -->

  <perspective>
    <layout>
      <!-- etc, etc -->
    </layout>
    <!-- etc, etc -->
  </perspective>
  <!-- etc, etc -->
</client:config>
```

✘ If Weave Hub is used in a multi-user environment, e.g. Citrix or Terminal Server, then user authentication *must* be enabled for users when running the Weave client.

Weave Hub Add-ins

Authority

Client PC Components

The Authority Weave Hub add-in is installed via a .msi file and only provides the option to change the installation directory, which defaults to `C:\Program Files\Cohga\WeaveHub\Authority\`. The installer also creates some registry entries to register the add-in with Weave Hub.

The Authority integration requires a program to be installed on the client PC which is called by the Authority add-in to initiate a request sent from Weave to Authority. This program should be provided by the Authority software vendor and setup by your IT department. The location of this program is provided by setting the `ulaunch` property in the Weave Authority configuration as shown below.

Partial Authority configuration sample showing ulaunch being set

```
<?xml version="1.0" encoding="UTF-8"?>

<config xmlns="urn:com.cohga.server.config#1.0"          xmlns:
authority="urn:com.cohga.weave.authority">

  <authority:config>
    <ulaunch>c:\Program Files\Authority\ulaunch.exe<
/ulaunch>
    <!-- other setting are also provided here -->
  </authority:config>

</config>
```

See the existing [Authority integration page](#) for information on what other settings can be included above.

Integration from Authority to Weave is now controlled by a new program called `AuthorityWeave.exe`, installed by the Weave Hub Authority add-in .msi file at `C:\Program Files\Cohga\WeaveHub\Authority\`, which when called will trigger Weave to perform the actions required by Authority. And the `AuthorityWeave.exe` should be called from the Authority program `Setup_customer.bat`.

Example Setup_customer.bat calling AuthorityWeave

```
@echo off

"C:\Program Files\Cohga\WeaveHub\Authority\AuthorityWeave.exe"
http://weave.lan:8080/weave/authority.html
```

The `AuthorityWeave.exe` program supports a single command line parameter which is the URL used to start Weave if it is not already running. If the parameter is not provided, by `Setup_customer.bat` for example, then the startup URL will be determined by the registry value `HKML\SOFTWARE\Cohga\WeaveHub\Authority\StartURL`. If this is not set, the Weave must already be running for the integration to work.

Weave Server Components

For the client to utilise the new Weave Hub integration, the `com.cohga.weave.authority` bundle must be updated, to at least version 2.0.0. The updated bundle provides the same functionality as the previous version of the Authority integration but uses the Weave Hub API to perform the communications.

Note that no changes are required in the client configuration to utilise the new integration. The client actions `com.cohga.authority.Send`, `com.cohga.authority.HandleNeighbourNotification`, etc, will determine if the new Weave Hub API is available and use that otherwise they will fall back to the older methods.

Pathway

Client PC Components

The Pathway Weave Hub add-in is installed via a .msi file and only provides the option to change the installation directory, which defaults to `C:\Program Files\Cohga\WeaveHub\Pathway\`. The installer also creates some registry entries to register the add-in with Weave Hub.

In addition to the Weave Hub add-in, the installer also installs and registers the component required for Pathway to communicate with Weave, `Pathway.COMGIS2`. The `Pathway.COMGIS2` component works by sending messages to Weave Hub, which Weave Hub then passes on to the Weave client for processing.

The Pathway integration components must also be installed on the client, this is supplied by Infor and should be located at `... \msc\Pathway.Integration.Setup.exe` at your Pathway installation location (which is generally a network share).

Weave Server Components

For the client to utilise the newer Weave Hub integration the `com.cohga.weave.pathway` bundle must be updated, to at least version 3.0.0. The updated bundle provides the same functionality as the previous version of the Pathway integration but uses the Weave Hub API to perform the communications. To utilise the updated actions in the client, the id of the actions need to be updated. Previously, for example, the action for sending the current selection to Pathway had the id of `com.cohga.pathway.Display`, and then `com.cohga.pathway2.Display` (when the actions were updated from using ActiveX to JavaScript). If you wish to utilise Weave Hub, the id will be `com.cohga.pathway3.Display`. Note that as was the case previously, the Pathway actions can be added directly to a toolbar or added within their own menu.

Adding some Pathway actions to the client

```
<client:config id="main">
  <title>Weave</title>
  <description>Main Weave Client</description>

  <plugin id="weave.hub"/> <!-- enable the weave.hub plugin -->
```

```

    <perspective>
      <layout>
        <!-- etc, etc -->
      </layout>

      <view ...>
        <toolbar>
          <!-- add as individual buttons to the
toolbar -->
          <item action="com.cohga.pathway3.
Display"/>
          <item action="com.cohga.pathway3.
SendQuery"/>
          <item action="com.cohga.pathway3.
DisplayOther"/>
          <item action="com.cohga.pathway3.
SendQueryOther"/>
          <item action="com.cohga.pathway3.
CreateLink"/>
          <item action="com.cohga.pathway3.
Return"/>

          <!-- add as a menu to the toolbar -->
          <item action="com.cohga.client.
actions.menuAction" text="" iconCls="icon-pathway_pathway">
            <item action="com.cohga.
pathway3.Display" text="Display"/>
            <item action="com.cohga.
pathway3.SendQuery" text="Send Query"/>
            <item action="com.cohga.
pathway3.DisplayOther" text="Display Other"/>
            <item action="com.cohga.
pathway3.SendQueryOther" text="Send Query Other"/>
            <item action="com.cohga.
pathway3.CreateLink" text="Create Link"/>
            <item action="com.cohga.
pathway3.Return" text="Return"/>
          </item>

          <!-- etc, etc -->
        </toolbar>
      </view>
    </perspective>
    <!-- etc, etc -->
  </client:config>

```

Pathway server configuration

```

<?xml version="1.0" encoding="UTF-8"?>

<config xmlns="urn:com.cohga.server.config#1.0" xmlns:pathway="urn:

```

```

com.cohga.pathway#1.0">

    <pathway:config>
        <database>Pathway_Demo</database>
        <link>
            <level>PROP</level>
            <entity>property</entity>
        </link>
    </pathway:config>

</config>

```

Pathway Configuration

Pathway must be configured to use the COM method of calling the GIS application and must use the `Pathway.COMGIS2` component to perform that integration, [which is outlined at the bottom of this page](#). `Pathway.COMGIS1` was used by Weave Link, and by using `Pathway.COMGIS2` it allows both the older Weave Link and the newer Weave Hub to be installed concurrently.

Unlike the Weave Link Pathway integration, the URL configured in *Pathway GIS Product Maintenance* and used to launch Weave, should just be the Weave URL. Weave Link required that the URL be prefixed with the title of the Window that Internet Explorer would display when the Weave client was running. This prefix is no longer required for Weave Hub and should be removed. For example:

```
Weave | http://gistest:8091/weave/main.html
```

would become just:

```
http://gistest:8091/weave/main.html
```

There is no executable required to be configured for the COMGIS based integration so you just need to set the *Path for Executable* in the *GIS Product Maintenance* to an existing file (as the form will not let you leave it blank).

TRIM/HP RM/HPE CM

Client PC Components

The Trim Weave Hub add-in is installed via a .msi file and only provides the option to change the installation directory, which defaults to `C:\Program Files\Cohga\WeaveHub\Trim\`. The installer also creates some registry entries, to register the add-in with Weave Hub.

In addition to the Weave Hub add-in, the installer also installs an executable, `TrimWeave.exe`, that can be used by Trim to communicate with Weave. The executable will be installed at the default location listed above unless it's changed during installation and can be used in the same way as the previous Weave Link version, [which is outlined at the bottom of this page](#). Like the Pathway integration, the executable just sends messages to Weave Hub which will then forward them on to the browser.

Weave Server Components

For the client to utilise the newer Weave Hub integration the `com.cohga.weave.trim` bundle must be updated, to at least version 2.0.0. The updated bundle provides the same functionality as the previous version of the Trim integration but uses the Weave Hub API to perform the communications. To utilise the updated actions in the client, the id of the actions need to be updated. Previously, for example, the action for sending the current selection to Trim has the id of `com.cohga.trim.Send`, and now if you wish to utilise Weave Hub the id will be `com.cohga.trim2.Send`.

Adding a Trim action to the client

```

<client:config id="main">
    <title>Weave</title>
    <description>Main Weave Client</description>

```

```

<plugin id="weave.hub"/> <!-- enable the weave.hub plugin -->

<perspective>
  <layout>
    <!-- etc, etc -->
  </layout>

  <view ...>
    <toolbar>
      <item action="com.cohga.trim2.Send"/>
<!-- send the selection to Trim -->
      <!-- etc, etc -->
    </toolbar>
  </view>

</perspective>
<!-- etc, etc -->
</client:config>

```

Proclaim

Client PC Components

The Proclaim Weave Hub add-in is installed via a .msi file and only provides the option to change the installation directory, which defaults to C:\Program Files\Cohga\WeaveHub\Proclaim\. The installer also creates some registry entries, to register the add-in with Weave Hub.

In addition to the Weave Hub add-in, the installer also installs and registers a COM object that can be used by Proclaim to communicate with Weave. The name of the COM object is `WeaveHubProclaim.MapAutomation` and must be registered within Proclaim as the component to call when communicating with a GIS, [which is outlined at the bottom of this page](#). Like the Pathway and Trim integrations, the COM object just sends messages to Weave Hub which will then forward them on to the browser.

Weave Server Components

For the client to utilise the newer Weave Hub integration the `com.cohga.weave.proclaim` bundle must be updated, to at least version 2.0.0. The updated bundle provides the same functionality as the previous version of the Proclaim integration but uses the Weave Hub API to perform the communications. To utilise the updated actions in the client, the id of the actions need to be updated. Previously, for example, the action for sending the current selection to Proclaim has the id of `com.cohga.proclaim.Send`, and now if you wish to utilise Weave Hub the id will be `com.cohga.proclaim2.Send`.

Adding a Proclaim action to the client

```

<client:config id="main">
  <title>Weave</title>
  <description>Main Weave Client</description>

  <plugin id="weave.hub"/> <!-- enable the weave.hub plugin -->

  <perspective>
    <layout>
      <!-- etc, etc -->
    </layout>

    <view ...>
      <toolbar>
        <item action="com.cohga.proclaim2.

```

```

Send"/> <!-- send the selection to Proclaim -->
                <!-- etc, etc -->
                </toolbar>
        </view>

        </perspective>
        <!-- etc, etc -->
</client:config>

```

Troubleshooting

The Weave Hub components generate logging output that can be captured using the [DebugView](#) tool from Microsoft. Having DebugView running while interacting with Weave Hub can capture output that may be useful in troubleshooting issues.

Weave Hub Service

The Weave Hub service can also be run directly, as opposed to being started via the Windows Services component, but make sure you stop the service before running the application directly. Running Weave Hub as a service allows you to see the log output generated as the application starts up, which may provide additional troubleshooting information, and also as the Weave Hub app communicates with the Weave Hub service.

Weave Hub App

The only log output for the Weave Hub App is via DebugView. It is not possible, like with the Weave Hub Service, to run the Weave Hub App as a console application to view the console output.

Sometimes it may be useful to stop the Weave Hub App so that missed log messages that occur during startup can be viewed via DebugView. But the Weave Hub App is intended to be running at all times when the user is logged in so, by default, there is no *Exit* entry in the Weave Hub App taskbar right-click menu. To enable an *Exit* menu item you can set a registry entry, note that if the Weave Hub App is installed there will already be a registry entry and the required location to show/hide the task bar icon (which can be edited to re-show the icon if it was hidden). The location where the registry entry should be created at is `HKLM\SOFTWARE\Cohga\WeaveHub\Core\`, the key name should be `ShowExit` and it should be set to a DWORD value of 1.

Browser

The Weave Hub plugin will output log messages to the browser console, so this is another location to look for output that may help resolve issues.

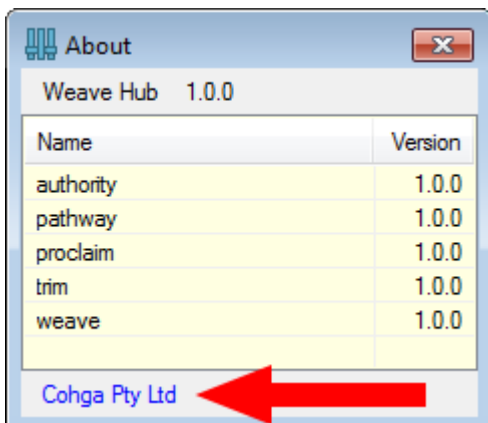
Microsoft Edge

Because of security restrictions Microsoft has implemented for Edge Weave Hub will not work unless the following command is run as an administrator on each client PC:

```
CheckNetIsolation LoopbackExempt -a -n=Microsoft.MicrosoftEdge_8wekyb3d8bbwe
```

What is the Weave Hub

This page is a placeholder for the link that will be opened when a user clicks on the Cohga Pty Ltd link in the About screen.



This content will be replaced with suitable content in the future.

Note that this page is redirected from <https://www.cohga.com/weavehub>