



BridgeLink Quick Start Guide Docker

For New Installations of BridgeLink 26.9.0

Table of Contents

1. About This Guide	3
2. Getting the Images	5
3. Setting Up the Deployment	6
4. Choosing and Connecting the Database	10
5. The Container Account	12
6. Starting BridgeLink for the First Time	14
7. Running WebAdmin	16
8. Finding the Server ID	19
9. Replacing the Certificates	21
10. The Data Pruner	25
11. Troubleshooting	28

CHAPTER 1

About This Guide

BridgeLink is Innovar Healthcare's integration platform. It moves messages between the systems in your organization and lets you build, run and watch those connections, called channels, from a web browser.

The steps here take one Linux host running Docker from an empty folder to a running BridgeLink 26.9.0 server, its database and WebAdmin.

An image is a packaged copy of a program and everything it needs to run. A container is one running copy of an image. Docker Compose starts several containers together from one settings file.

If you would like a hand with any part of the install, Innovar Healthcare's professional services team can help with as much or as little of it as you want.

What You Run

Container	What it is	Runs as	Port
b1	The BridgeLink server, with Java already inside the image	A non-root user inside the container	8443 (HTTPS)
postgres	The database BridgeLink keeps its channels, messages, events, users and settings in. You can point BridgeLink at an existing database server instead.	The database image's own user	Not published outside Docker
webadmin	The browser-based tool you use to manage BridgeLink, in its own container	A non-root user inside the container	8444 (HTTPS)

Open port **8443** on the BridgeLink server to other machines only if WebAdmin runs on another machine or other programs call the BridgeLink API. Channels that listen for connections, such as MLLP or HTTP listeners, need their own ports opened. Port **8080** is BridgeLink's plain-HTTP port; nothing in this guide uses it, so keep it closed to other machines.

The Compose file in [Setting Up the Deployment](#) already publishes **8443** on every interface of the Docker host, and Docker-published ports bypass host firewalls such as **ufw**.

WebAdmin reaches BridgeLink over the Docker network and does not need it. To keep **8443** local to the Docker host, change "8443:8443" to "127.0.0.1:8443:8443", and use **localhost** in step 9 of [Replace the BridgeLink Server Certificate](#). A channel that listens for

connections also needs its port added under **ports:** for **b1**.

With Docker there is no installer to run and no archive to unpack: each image already holds the software and everything it needs. Your choices are which image variant to use and the settings you pass in.

Requirements

- A Linux host, on **linux/amd64** or **linux/arm64**, with Docker Engine and the Docker Compose plugin. The plugin is version 2 of Compose, which you run as **docker compose** (with a space, not **docker-compose**). To install both, follow Docker's instructions at <https://docs.docker.com/engine/install/>.
- Permission to run **docker** commands. See **Run Docker Without Root** below.
- Internet access to Docker Hub, where the BridgeLink images are published
- **sudo** rights on the host, to set folder ownership
- To replace the BridgeLink server certificate (**Replacing the Certificates**): **keytool** from a Java 17 or 21 runtime on the host
- A current browser for WebAdmin: Chrome, Edge or Firefox 120 or later, or Safari 17 or later

Run Docker Without Root

On a new Linux host, the first **docker** command fails with **permission denied while trying to connect to the docker API at unix:///var/run/docker.sock**. Your account is not allowed to use Docker yet. You have two ways to fix that:

- Add your account to the **docker** group, then sign out and sign back in so the change takes effect
- Put **sudo** in front of every **docker** command in this guide

To add your account to the **docker** group, run this in a terminal on the Linux host:

```
sudo usermod -aG docker $USER
```

Docker Desktop. This guide is written for a Linux host. If you try it on Docker Desktop on a Mac or Windows computer, skip the **chown** commands in this guide, unless your project folder is inside a WSL2 Linux filesystem, where files keep their real ownership. See **Setting Up the Deployment**.

CHAPTER 2

Getting the Images

BridgeLink server images are on Docker Hub, in the `innovarhealthcare/bridgelink` repository. You can pull them without signing in to Docker Hub.

Choose an Image Variant

Each variant is a tag of the same repository:

Image	What it is	Runs as
<code>innovarhealthcare/bridgelink:26.9.0</code>	The standard image, built on Rocky Linux 9	User ID 1000
<code>innovarhealthcare/bridgelink:26.9.0-dhi</code>	The hardened image, built on a minimal Docker Hardened Image base with no package manager and no command-line tools. It is rebuilt every week to pick up operating system security fixes.	User ID 65532
<code>innovarhealthcare/bridgelink:26.9.0-dhi-slim</code>	The hardened image without the desktop Administrator client, for sites that manage BridgeLink only through WebAdmin	User ID 65532

Each tag also has a Java 17 form with `-jdk17` added at the end, such as `26.9.0-jdk17`, `26.9.0-dhi-jdk17` and `26.9.0-dhi-slim-jdk17`. Unless you need Java 17, use the tags in the table.

The two hardened images run BridgeLink the same way as the standard one. They differ in the base system, the user ID (65532), having no shell, and writing `MP_` values literally; the slim image also leaves out the Administrator client. The examples in this guide use the standard image. [The Container Account](#) lists the two changes for a hardened image.

Pull the Image

```
docker pull innovarhealthcare/bridgelink:26.9.0
```

Pin by digest for production. The hardened tags, such as `26.9.0-dhi`, are rebuilt every week with security fixes, and tags such as `26.9` and `latest` move to newer releases, so the image behind a tag can change. When you need an image that never changes, pull it by digest, a fixed fingerprint of one exact image (`innovarhealthcare/bridgelink@sha256:...`).

CHAPTER 3

Setting Up the Deployment

Everything lives in one folder on the host: a Compose file and the `appdata` folder BridgeLink keeps its keystore and server ID in.

1. Create the folder:

```
mkdir bridgelink
```

2. Go into it:

```
cd bridgelink
```

3. Create the `appdata` folder:

```
mkdir appdata
```

4. Give it to the container's user, ID `1000` for the standard image:

```
sudo chown 1000:1000 appdata
```

Docker Desktop (Mac or Windows). Skip this step and every other `chown` step in this guide. Docker Desktop reads a mounted folder as your own desktop user, so giving the folder to user `1000` locks BridgeLink out of it. The exception is a project folder inside a WSL2 Linux filesystem, where files keep their real ownership, so run the `chown` steps there.

5. Read the WebAdmin license terms: the Business Source License 1.1 and the BridgeLink WebAdmin Supplemental Terms. To read them without starting WebAdmin:

```
docker run --rm --entrypoint cat innovarhealthcare/bridgelink-webadmin:26.9.0 \
/app/LICENSE /app/SUPPLEMENTAL-TERMS.md
```

6. To accept them, create a file named `.env` in the `bridgelink` folder with this one line. Compose reads it every time, so WebAdmin starts on every later command too:

```
echo BL_ACCEPT_LICENSE=1 > .env
```

7. In the `bridgelink` folder, create a file named `docker-compose.yml` with the content below, for example with `nano docker-compose.yml`.

YAML depends on its indentation. If you are reading the PDF, do not copy the content from it: the copy loses the indentation and picks up page headers and footers. Copy it from the HTML version of this guide instead, or use the `docker-compose.yml` file that comes with this guide.

```
services:
  bl:
    image: innovarhealthcare/bridgelink:26.9.0
    user: "1000:1000"
    ports:
      - "8443:8443"
    volumes:
      - ./appdata:/opt/bridgelink/appdata
    environment:
      MP_DATABASE: postgres
      MP_DATABASE_URL: jdbc:postgresql://postgres:5432/bridgelinkdb
      MP_DATABASE_USERNAME: bridgelink
      MP_DATABASE_PASSWORD: change-me-database
      MP_KEYSTORE_STOREPASS: change-me-store
      MP_KEYSTORE_KEYPASS: change-me-key
    security_opt:
      - no-new-privileges:true
    stop_grace_period: 35s
    depends_on:
      - postgres
    restart: unless-stopped

  postgres:
    image: postgres:16-alpine
    environment:
      POSTGRES_DB: bridgelinkdb
      POSTGRES_USER: bridgelink
      POSTGRES_PASSWORD: change-me-database
    volumes:
      - pgdata:/var/lib/postgresql/data
    restart: unless-stopped

  webadmin:
    image: innovarhealthcare/bridgelink-webadmin:26.9.0
    ports:
      - "8444:8444"
    environment:
      PORT: "8444"
      NODE_ENV: production
      BRIDGELINK_SERVER_URL: https://bl:8443
      BL_ACCEPT_LICENSE: ${BL_ACCEPT_LICENSE:-}
    volumes:
      - webadmin-certs:/app/certs
    security_opt:
      - no-new-privileges:true
    depends_on:
      bl:
        condition: service_healthy
    restart: unless-stopped
```

```
volumes:
  pgdata:
  webadmin-certs:
```

8. In `docker-compose.yml`, replace each `change-me-...` value with a password of your own. Follow these rules:

- Replace the whole value, including its `-database`, `-store` or `-key` ending
 - Use letters and digits only. Compose treats a `$` as the start of a variable. With the standard image, a `|` in `MP_DATABASE_PASSWORD` cannot be written, so BridgeLink starts with an empty database password, cannot sign in to the database, and restarts again and again (see [Troubleshooting](#)). A backslash is unsafe in every password with both images, because BridgeLink reads `mirth.properties` with backslash as an escape character.
 - Use one password for the database and a different one for each keystore value
 - Under `bl:`, set `MP_DATABASE_PASSWORD`, `MP_KEYSTORE_STOREPASS` and `MP_KEYSTORE_KEYPASS`
 - Under `postgres:`, set `POSTGRES_PASSWORD` to the same value as `MP_DATABASE_PASSWORD`
- What the settings do:

Setting	Why it is there
<code>./appdata:/opt/bridgelink/appdata</code>	Keeps the keystore and the server ID on the host, so they survive when the container is replaced
<code>MP_DATABASE</code> and the other <code>MP_DATABASE...</code> lines	Point BridgeLink at the <code>postgres</code> container. See Choosing and Connecting the Database .
<code>MP_KEYSTORE_STOREPASS</code> and <code>MP_KEYSTORE_KEYPASS</code>	Fix the keystore passwords. The keystore lives in <code>appdata</code> and must open with the same passwords every time the container is created.
<code>no-new-privileges:true</code>	Stops anything in the <code>bl</code> or <code>webadmin</code> container from gaining extra Linux privileges after it starts
<code>stop_grace_period: 35s</code>	Lets BridgeLink shut down cleanly when the container stops. Messages still queued stay queued.
<code>BL_ACCEPT_LICENSE: \${BL_ACCEPT_LICENSE:-}</code>	Takes your acceptance of the WebAdmin license terms from the <code>.env</code> file. The Compose file does not accept them for you. See Running WebAdmin .

Setting	Why it is there
<code>condition: service_healthy</code>	Starts WebAdmin only once BridgeLink reports healthy, so the first sign-in does not reach BridgeLink before it has finished starting

Keep passwords out of the file for production. For a production host, move the passwords into a Docker secret. BridgeLink reads a secret named `mirth_properties` and merges the properties in it, such as `database.password` and the two keystore passwords, into its settings.

Run every `docker compose` command in this guide from the `bridgeline` folder. From any other folder, the commands fail with `no configuration file provided: not found`.

CHAPTER 4

Choosing and Connecting the Database

Every BridgeLink server keeps its channels, messages, events, users and settings in one database. BridgeLink reads that choice from its settings file, `conf/mirth.properties`. In a container you set those settings with environment variables, and the container writes them into `mirth.properties` each time it starts.

Always Use an External Database in a Container

BridgeLink ships set to the default Derby database. Do not use it in a container:

- **Cannot run on the `-jdk17` tags.** BridgeLink 26.9 will not start a server on the default Derby database with any Java older than 21. The standard images use Java 21, but the tags ending in `-jdk17` do not, and on those the server stops at once and logs `embedded Derby requires Java 21+ as of 26.9; upgrade Java or switch to an external database`.
- **Runs inside BridgeLink.** Derby lives in the same process as the server, so no other tool can open the database while BridgeLink runs, and its files would have to live in a volume and be backed up by copying.
- **Lacks features.** The default Derby database does not support the JSON value type in **Lookup Manager** or regex search in **Message Browser**. SQL Server does not support regex search either.
- **Sized for evaluation.** For production message volumes, use PostgreSQL, MySQL, Microsoft SQL Server or Oracle.

The Compose file in [Setting Up the Deployment](#) runs PostgreSQL next to BridgeLink. To use a database server you already run, delete the `postgres` service and point the variables below at your server.

The Database Variables

Variable	What to set
MP_DATABASE	postgres, mysql, sqlserver or oracle
MP_DATABASE_URL	The connection address, using the pattern below
MP_DATABASE_USERNAME	The BridgeLink login
MP_DATABASE_PASSWORD	That login's password

Database	MP_DATABASE_URL pattern
PostgreSQL	<code>jdbc:postgresql://dbhost:5432/bridgelinkdb</code>
MySQL	<code>jdbc:mysql://dbhost:3306/bridgelinkdb</code>

Database	MP_DATABASE_URL pattern
Microsoft SQL Server	<code>jdbc:sqlserver://dbhost:1433;databaseName=bridgelinkdb</code>
Oracle	<code>jdbc:oracle:thin:@//dbhost:1521/serviceName</code> for a service name (pluggable databases), or <code>jdbc:oracle:thin:@dbhost:1521:SID</code> for a SID

SQL Server and certificates. BridgeLink 26.9 connects to SQL Server with encryption on and checks the server's certificate. If your SQL Server's certificate is self-signed, comes from an internal certificate authority, or does not match the host name in the URL, BridgeLink cannot connect. Either add that certificate, or its certificate authority, to the Java trust store BridgeLink uses, or add `;trustServerCertificate=true` to the end of the URL. That keeps the connection encrypted but does not check which server answered. In a container, adding a certificate to Java's trust store means building your own image or mounting a trust store and pointing to it with `MP_VMOPTIONS (-Djavax.net.ssl.trustStore=...)`, so `;trustServerCertificate=true` is the simpler route.

For an existing database server, first create an empty database for BridgeLink and a login that owns it and can create tables. BridgeLink creates all of its tables the first time it starts.

Setting Any Other Property

Any variable that starts with `MP_` sets the matching line in `mirth.properties`. Drop the `MP_` prefix, turn each `__` into `-`, turn each remaining `_` into `.`, and change every letter to lowercase. For example, `MP_DATABASE__READONLY__MAX__CONNECTIONS` sets `database-readonly.max-connections`.

Because the name is lowercased, you cannot set a property whose name has a capital letter this way. The one exception is `MP_SERVER_ALLOWROOT`.

Two other variables do not follow this rule. `MP_VMOPTIONS` writes the `vmoptions` file rather than a line in `mirth.properties`, as [Memory for the BridgeLink Server](#) describes, and `MP_DATABASE_RETRY_WAIT` sets `database.connection.retrywaitinmilliseconds`, the wait in milliseconds between database connection attempts at startup.

CHAPTER 5

The Container Account

In a container, BridgeLink runs as the user the container runs as. That user decides what BridgeLink can write to, the job a service account does on a server.

Why Not Root

Channels can run scripts. If BridgeLink runs as **root**, any script in any channel has full control of the container. So BridgeLink refuses to start as **root** by default, and the log shows:

ERROR: BridgeLink is running as root/Administrator.

The BridgeLink images already run as a non-root user:

Image	User ID
Standard (tag 26.9.0)	1000
Hardened (tags 26.9.0-dhi and 26.9.0-dhi-slim)	65532

Keep the **user:** line in the Compose file matched to your image, and never set it to **root** or **0**.

Any host folder you mount into the BridgeLink container, such as **appdata**, must belong to that user ID so BridgeLink can write to it.

Using a Hardened Image

Make these two changes to **Setting Up the Deployment**:

1. Give **appdata** to user ID 65532 instead of 1000:

```
sudo chown -R 65532:65532 appdata
```

Docker Desktop (Mac or Windows). Skip this step, unless your project folder is inside a WSL2 Linux filesystem, where files keep their real ownership. See **Setting Up the Deployment**.

2. In **docker-compose.yml**, change the **image:** and **user:** lines under **bl:** so they read:

```
services:
  bl:
    image: innovarhealthcare/bridgeline:26.9.0-dhi
    user: "65532:65532"
```

A hardened container has no shell and no command-line tools, so **docker compose exec bl sh** fails with **executable file not found**. To check a hardened container, read its log with

`docker compose logs b1` and look at the files in `appdata` on the host.

Changing the User After Setup

Suppose the deployment ran with one image and you switch to the other, or you change the `user:` line. The files in `appdata` still belong to the old user ID. The new user cannot write the keystore or its other files, and BridgeLink fails with **Permission denied** errors.

1. Stop BridgeLink:

```
docker compose stop b1
```

2. Give `appdata` to the new user ID, for example `65532`:

```
sudo chown -R 65532:65532 appdata
```

3. In `docker-compose.yml`, change the `image:` and `user:` lines under `b1:` to the new image and user ID. For the hardened image, they read:

```
services:
  b1:
    image: innovarhealthcare/bridgeline:26.9.0-dhi
    user: "65532:65532"
```

4. Give the new user ID the same access to any other host folders you mount for your channels.
5. Start BridgeLink with the new settings:

```
docker compose up -d b1
```

CHAPTER 6

Starting BridgeLink for the First Time

1. From the `bridgelink` folder, start everything:

```
docker compose up -d
```

`docker compose up -d` does not return until `b1` reports healthy, because WebAdmin waits for it. On a normal start that takes about a minute or less. If the command has not returned after a couple of minutes, open a second terminal in the `bridgelink` folder, run step 2 to see what `b1` is doing, and see [Troubleshooting](#).

2. Follow the BridgeLink log:

```
docker compose logs -f b1
```

3. Wait for the newest line that contains `Web server running at`, with a time after the start you just made. That line is the sign that BridgeLink started. Each log line begins with the container name, the log level and a time, so look for the text anywhere in the line.
4. Find the newest line that contains `Running`, to check the database name and Java version. Confirm it shows `postgres`, not `derby`.
5. Press `Control-C` to stop following the log. The containers keep running.
6. Check the containers:

```
docker compose ps
```

The `b1` container reports `healthy` once BridgeLink has started and finished deploying its channels. A first start can take longer on a slow host or a host that is busy with other work.

A `Running` line on its own does not mean BridgeLink started. If the keystore cannot be opened, or `appdata` cannot be written, the log still shows `Running`, but it never shows `Web server running at`, and port `8443` refuses connections. See [Troubleshooting](#).

On this first start, BridgeLink:

- Creates its tables in the database
- Creates a keystore in `appdata` holding a new self-signed certificate for port `8443`
- Creates `appdata/server.id`, which holds this server's unique ID

If PostgreSQL is still starting when BridgeLink first tries it, BridgeLink retries twice, 10 seconds apart, then stops. `restart: unless-stopped` starts it again, so give it a minute before you look for a problem.

Memory for the BridgeLink Server

BridgeLink starts with a 256 MB Java heap, which is enough for a personal or test install but small for production. The release's `blserver.vmoptions` sets it.

To change it, add `MP_VMOPTIONS` at the end of the `environment:` section under `bl:` in `docker-compose.yml`. A bare number sets the heap in MB:

```
services:
  bl:
    environment:
      MP_KEYSTORE_KEYPASS: your-keystore-key-password
      MP_VMOPTIONS: "1024"
```

Leave memory for the host and for the database. If you give the container a memory limit, the limit must be larger than the heap. To apply the change, run this from the `bridgelink` folder:

```
docker compose up -d bl
```

CHAPTER 7

Running WebAdmin

WebAdmin is the browser-based tool you use to manage BridgeLink. In Docker it is always a separate container: no BridgeLink image includes it. The Compose file in [Setting Up the Deployment](#) already runs it.

The WebAdmin Container

Setting	What it does
<code>image: innovarhealthcare/bridgelink-webadmin:26.9.0</code>	The WebAdmin image. It also includes the screens for commercial plugins such as SSL Settings, OIDC and Access Control, so you need no other image.
<code>BRIDGELINK_SERVER_URL: https://b1:8443</code>	Locks WebAdmin to the b1 container. The address must start with <code>https://</code> .
<code>BL_ACCEPT_LICENSE: \${BL_ACCEPT_LICENSE:-}</code>	Passes on your acceptance of the Business Source License 1.1 (the license WebAdmin is released under) and the BridgeLink WebAdmin Supplemental Terms, from the <code>BL_ACCEPT_LICENSE=1</code> line in <code>.env</code> . Read both before you accept. Without the <code>.env</code> line, the container prints a short notice that points to the terms, not the terms themselves, and stops. Because of <code>restart: unless-stopped</code> it keeps restarting, at first every few seconds and then about once a minute, printing the notice each time.
<code>webadmin-certs:/app/certs</code>	Keeps WebAdmin's certificate when the container is replaced

WebAdmin reaches BridgeLink over the Docker network by the service name `b1`.

Sign In for the First Time

1. Open `https://your-docker-host:8444` in a browser.
2. If the browser shows a certificate warning, click **Advanced**, then **Proceed to ... (unsafe)** in Chrome or **Continue to ... (unsafe)** in Edge. In Firefox, click **Advanced**, then **Accept the Risk and Continue**. Read the note after this step.

When a browser stops letting you click through. WebAdmin tells browsers to accept only a valid certificate for its server name and every name under it, on every port, for two years. This is called HSTS. A browser starts enforcing it the first time it opens WebAdmin with a certificate it trusts, for example after you install your own certificate (see [Replacing the Certificates](#)). Clicking through a warning does not start it. `localhost` and IP addresses are never affected. Once a browser enforces HSTS for a name, it also refuses BridgeLink's default certificate on port 8443 at that name.

3. In **Username**, type `admin`.
4. In **Password**, type `admin`.
5. Click **Login**.
6. On **Welcome to BridgeLink**, leave **Username** as `admin`.
7. In **New Password**, type a new password of at least 8 characters, with an uppercase letter, a lowercase letter, a number, and a special character such as `!`, `@`, `#`, `$`, `%` or `*`. A hyphen, period or space does not count as a special character, and the password cannot be `admin`. WebAdmin lists only the rules that `admin` breaks, so check your password against all of these.
8. Type it again in **Confirm Password**.
9. The profile fields on **Welcome to BridgeLink** are optional; fill them in if you want to.
10. Click **Save & Continue**.

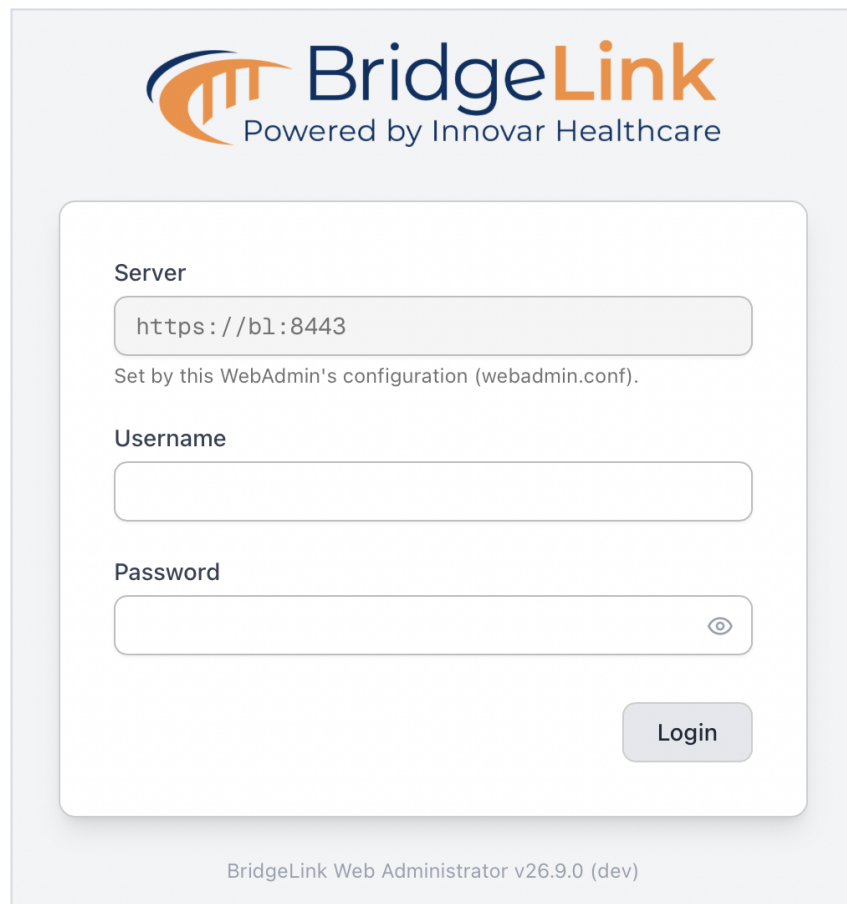
The image shows a web browser window displaying the BridgeLink Web Administrator sign-in interface. At the top, the BridgeLink logo is shown with the tagline "Powered by Innovar Healthcare". Below the logo is a white rounded rectangle containing the login form. The form has three input fields: "Server" with the value "https://bl:8443", "Username" which is empty, and "Password" which is empty and has a toggle icon on the right. Below the "Server" field is a note: "Set by this WebAdmin's configuration (webadmin.conf)". At the bottom right of the form is a "Login" button. Below the form, the text "BridgeLink Web Administrator v26.9.0 (dev)" is displayed.

Figure 1: The WebAdmin sign-in screen in a browser, showing the Server field filled in with `https://bl:8443` and read-only, with the note "Set by this WebAdmin's configuration (webadmin.conf)." under it, the Username and Password fields, and the Login button.

Controlling the WebAdmin Container

WebAdmin runs in its own container, so you control it with Docker Compose from the `bridgelink` folder. WebAdmin's errors go to its container log:

```
docker compose logs webadmin
```

To restart WebAdmin:

```
docker compose restart webadmin
```

More WebAdmin Setups

For WebAdmin outside Docker, one WebAdmin for several servers, your own certificate, and every setting, see the **BridgeLink WebAdmin Setup Guide** on the BridgeLink download page, <https://www.bridgelink.net/download>.

CHAPTER 8

Finding the Server ID

Every BridgeLink server has a unique server ID, created the first time it starts. Plugin licenses are issued against this ID, so Innovar Healthcare asks for it when you license a plugin.

In WebAdmin

1. Sign in to WebAdmin.
2. In the left sidebar, scroll to the bottom and, under **Help**, click **About BridgeLink**.
If **Update Available** opens instead, click **Got it**, then click **About BridgeLink** again.
3. Click the copy icon beside **Server ID**.

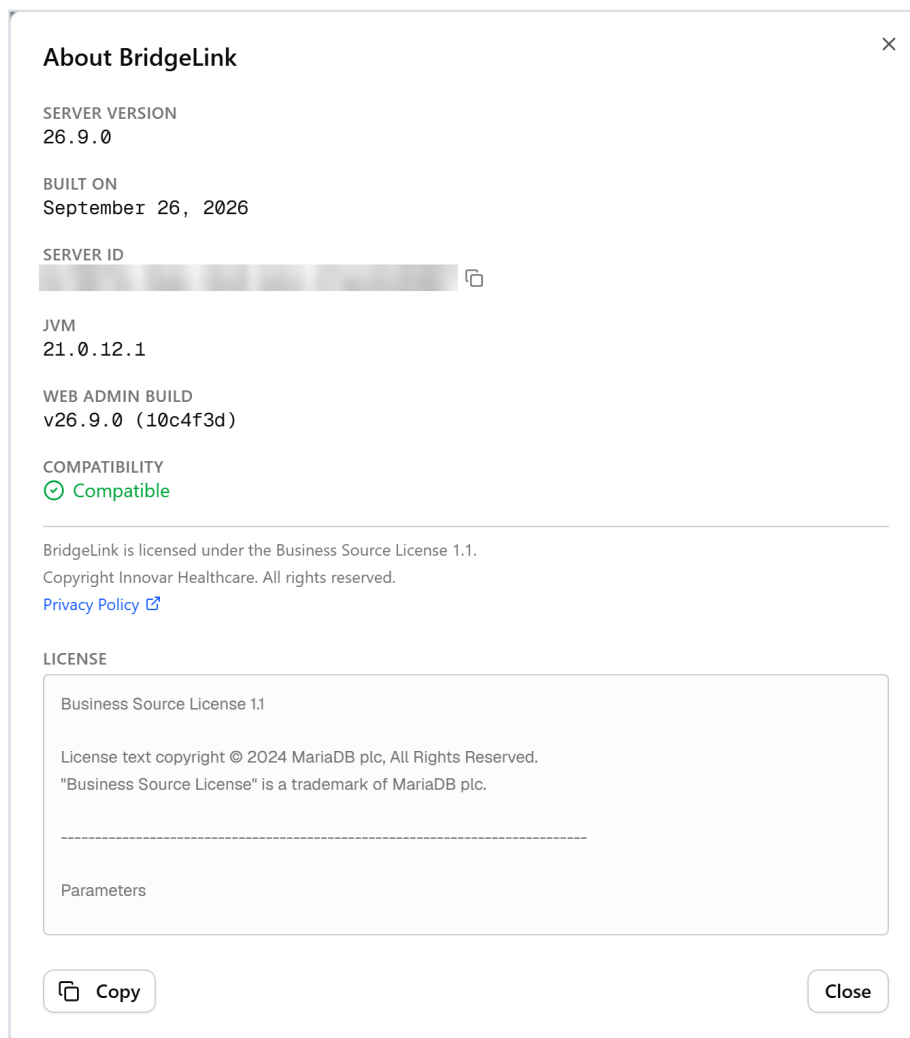


Figure 2: The About BridgeLink dialog in WebAdmin, showing the Server Version, Built On, Server ID (blurred here, with its copy icon), JVM, Web Admin Build and Compatibility rows.

On the Host

The same ID is in the `appdata` folder you mounted:

```
sudo cat appdata/server.id
```

The file holds one line in the form `server.id = <ID>`. The ID is the part after the equals sign.

Back up the whole `appdata` folder together with your `docker-compose.yml` (and `.env`, if you use one). Besides `server.id` and `keystore.jks`, `appdata` holds the Configuration Map. The keystore holds the key BridgeLink uses to encrypt data, and the Compose file holds the keystore passwords. The database is separate: channels, users, settings and messages live in the `pgdata` volume, so back up the database too, for example with `pg_dump`. If you use the `mirth_properties` secret, back up that file as well. Settings changed at runtime in `mirth.properties` are lost when the container is recreated; set them with `MP_` variables instead.

Keep the ID Fixed

A container is easy to replace, and a replaced container without its `appdata` folder would get a new ID. To make sure the ID never changes, set it in the Compose file once you have it:

1. Copy the ID from **About BridgeLink**.
2. In `docker-compose.yml`, find the `bl:` service and the `environment:` section inside it, the one with the `MP_` lines. Add this line at the end of that section, lined up with the `MP_` lines, with your ID in place of `your-server-id`. Keep the indentation exactly as shown, because YAML depends on it:

```
services:
  bl:
    environment:
      MP_KEYSTORE_KEYPASS: your-keystore-key-password
      SERVER_ID: your-server-id
```

3. Apply the change, from the `bridgelink` folder:

```
docker compose up -d bl
```

The container writes this ID into `appdata/server.id` every time it starts.

CHAPTER 9

Replacing the Certificates

A new deployment uses two self-signed certificates (certificates the server creates for itself, which browsers do not trust), which is why browsers warn the first time. Before you use BridgeLink in production, replace them with certificates from your organization's certificate authority (CA).

Certificate	Port	Who sees it	Where it lives
WebAdmin	8444	Every browser that opens WebAdmin	/app/certs/server.crt and server.key in the webadmin container
BridgeLink server	8443	WebAdmin and any program that calls the BridgeLink API	The mirthconnect entry in appdata/keystore.jks

This chapter has two separate procedures, one for each certificate. Each has its own numbered steps, and you do not need to do one to do the other.

Replace the WebAdmin Certificate

Before you replace WebAdmin's certificate, read **When a browser stops letting you click through** under **Sign In for the First Time** in **Running WebAdmin**.

You need the certificate and its private key as PEM files (text files that begin with -----BEGIN). The certificate must be one that users' browsers trust, and it must name the address they type to reach WebAdmin. Put any intermediate certificates in the certificate file after the server certificate. The private key file must not have a password.

1. Create a folder on the host for them:

```
sudo mkdir -p /etc/ssl/bridgelinek-webadmin
```

2. Copy your certificate into it as `server.crt`. Replace `/path/to/your-certificate.pem` with the path to your own certificate file:

```
sudo cp /path/to/your-certificate.pem /etc/ssl/bridgelinek-webadmin/server.crt
```

3. Copy your private key into it as `server.key`. Replace `/path/to/your-private-key.pem` with the path to your own key file:

```
sudo cp /path/to/your-private-key.pem /etc/ssl/bridgelinek-webadmin/server.key
```

4. Give the folder to the WebAdmin container's user, ID `1000`:

```
sudo chown -R 1000:1000 /etc/ssl/bridgelink-webadmin
```

5. Make the private key readable only by its owner:

```
sudo chmod 600 /etc/ssl/bridgelink-webadmin/server.key
```

6. In `docker-compose.yml`, find the `webadmin:` service. Replace its existing `- webadmin-certs:/app/certs` line with the line below, and leave the other `volumes:` sections as they are. Include the `-` at the start of the line, because it is part of the file. Keep the indentation exactly as shown, because YAML depends on it:

```
webadmin:
  volumes:
    - /etc/ssl/bridgelink-webadmin:/app/certs:ro
```

7. Recreate the WebAdmin container, from the `bridgelink` folder:

```
docker compose up -d webadmin
```

8. Open WebAdmin in a browser and confirm there is no certificate warning.

To Undo the Swap

If browsers cannot open WebAdmin after the swap, put the old certificate back:

1. In `docker-compose.yml`, put the `- webadmin-certs:/app/certs` line back in place of the line you added in step 6.
2. Recreate the WebAdmin container, from the `bridgelink` folder:

```
docker compose up -d webadmin
```

The original certificate returns from the `webadmin-certs` volume.

Browsers that opened WebAdmin by its server name with the new certificate refuse the old self-signed one. Fix the new certificate instead, or clear the browser's HSTS entry for the server name: open `chrome://net-internals/#hsts` in Chrome or `edge://net-internals/#hsts` in Edge, and use **Delete domain security policies**. In other browsers, clear the browsing data for that server name.

Replace the BridgeLink Server Certificate

BridgeLink keeps its certificate in a keystore file under the name `mirthconnect`. The same keystore also holds the key BridgeLink uses to encrypt data, so replace only the `mirthconnect` entry. Never swap in a different keystore file.

You need the certificate and private key as a PKCS12 file (a single file, ending in `.p12` or `.pfx`, that bundles a certificate with its private key) and its password, plus `keytool` on the host,

which comes with any Java 17 or 21 install.

The commands below contain placeholders in angle brackets. Replace each one, including the angle brackets, with your own value:

Placeholder	Where the value comes from
<your MP_KEYSTORE_STOREPASS value>	The MP_KEYSTORE_STOREPASS value under b1: in your docker-compose.yml
<your MP_KEYSTORE_KEYPASS value>	The MP_KEYSTORE_KEYPASS value under b1: in your docker-compose.yml
<your .p12 file password>	The password set on your PKCS12 file when it was created, which you get from whoever issued or exported the file
<the entry name from step 4>	The name that keytool lists for the entry in your PKCS12 file in step 4

1. Stop BridgeLink:

```
docker compose stop b1
```

2. Back up the keystore:

```
sudo cp appdata/keystore.jks keystore.jks.bak
```

3. Copy your PKCS12 file into the **bridgeline** folder as **server.p12**. The commands that follow run from that folder and expect the file there.

4. List the entry name inside your PKCS12 file:

```
keytool -list -keystore server.p12 -storetype PKCS12
```

5. Remove the old certificate entry:

```
sudo keytool -delete -alias mirthconnect -keystore appdata/keystore.jks \
  -storetype JCEKS -storepass <your MP_KEYSTORE_STOREPASS value>
```

6. Import yours as mirthconnect:

```
sudo keytool -importkeystore -srckeystore server.p12 -srcstoretype PKCS12 \
  -srcstorepass <your .p12 file password> \
  -srcalias <the entry name from step 4> \
  -destkeystore appdata/keystore.jks -deststoretype JCEKS \
  -deststorepass <your MP_KEYSTORE_STOREPASS value> -destalias mirthconnect \
  -destkeypass <your MP_KEYSTORE_KEYPASS value>
```

Steps 5 and 6 print a warning that recommends migrating the keystore to PKCS12. Ignore it: BridgeLink reads the keystore as JCEKS, so do not convert it.

7. Give the keystore back to the container's user, ID **1000** for the standard image:

```
sudo chown 1000:1000 appdata/keystore.jks
```

For a hardened image, which runs as user **65532**, run this instead:

```
sudo chown 65532:65532 appdata/keystore.jks
```

8. Start BridgeLink:

```
docker compose start bl
```

9. Open <https://your-docker-host:8443> in a browser and confirm the browser shows your certificate.

If BridgeLink does not start, or port 8443 does not answer: stop BridgeLink, copy the backup `keystore.jks.bak` you made in step 2 back to `appdata/keystore.jks`, repeat the `chown` from step 7, and start BridgeLink.

WebAdmin keeps working through the change, with no change to its own settings.

CHAPTER 10

The Data Pruner

BridgeLink keeps the messages each channel processes until the Data Pruner removes them. The Data Pruner is off on a new server, so turn it on before you send real traffic through BridgeLink; otherwise the database grows with every message.

It has two parts, and both have to be set:

- **The schedule, for the whole server:** in WebAdmin, in the left sidebar under **Administration**, click **Settings**, then the **Data Pruner** tab
- **How long each channel keeps messages:** on each channel's **Summary** tab, under **Message Pruning**. A channel whose pruning fields are blank is never pruned, even when the schedule is on.

Choose ages that match your organization's record-keeping rules.

Turn On the Schedule

1. In WebAdmin, in the left sidebar under **Administration**, click **Settings**.
2. Click the **Data Pruner** tab.
3. Under **Schedule**, set **Enable** to **Yes**.
4. Choose a **Schedule Type** and set when it runs: **Interval** runs every set number of hours or minutes, **Time** runs once a day, and **Cron** takes cron expressions. Once a day at a quiet hour suits most servers.
5. Click **Save**.

Settings

Server Administrator Tags Configuration Map Resources **Data Pruner** Message Trends Version History

Status

Current State: **Not running**

Current Process: -

Last Process: -

Next Process: Scheduled Monday, Sep 28, 1:00:00 AM

Schedule

Enable: ☒ Yes ☐ No

Schedule Type: **Time**

Time (HH : MM): **1** : **0**

Prune Settings

Block Size: **1000** (50-10000, recommended: 1000)

Prune Events: ☒ Yes ☐ No

Prune Event Age: **90** days

Archive Settings

Enable Archiving: ☐ Yes ☒ No

Figure 3: The Data Pruner tab of the Settings page, showing the Status section, the Schedule section with Enable set to Yes and Schedule Type set to Time, the Prune Settings section and the Archive Settings section.

Set How Long Each Channel Keeps Messages

1. In the left sidebar under **Build**, click **Channels**.
2. Double-click the channel to open it in the channel editor.
3. On the **Summary** tab, under **Message Pruning**, enter the number of days in **Prune metadata after** and **Prune content after**. Content cannot be kept longer than metadata.
4. Click **Save**.
5. Repeat for each channel.

Message Pruning

Prune metadata after **30** days

Prune content after **7** days

☒ Archive messages before pruning

☐ Prune errored messages

(incomplete, errored, and queued messages will not be pruned)

Figure 4: The Message Pruning section of a channel's Summary tab, showing Prune metadata after set to 30 days, Prune content after set to 7 days, Archive messages before pruning checked and Prune errored messages cleared.

Incomplete and queued messages are never pruned. Errored messages are kept too, unless you check **Prune errored messages**.

CHAPTER 11

Troubleshooting

Symptom	Likely cause	What to do
The b1 container on a -jdk17 tag exits at once, and <code>docker compose logs b1</code> shows "embedded Derby requires Java 21+ as of 26.9; upgrade Java or switch to an external database"	MP_DATABASE is missing, so BridgeLink fell back to the default Derby database, which the -jdk17 tags cannot run	Set the four MP_DATABASE variables. See The Database Variables . The standard images do not exit in this case: they start quietly on Derby, so check that the Running log line shows postgres.
The b1 container exits at once, and the log shows "ERROR: BridgeLink is running as root/Administrator."	The user: line was set to root or 0	Set user: to "1000:1000" for the standard image or "65532:65532" for a hardened one.
The command <code>docker compose up -d</code> does not finish. <code>docker compose ps -a</code> shows b1 as health: starting and webadmin as Created, and <code>docker compose logs b1</code> shows "Permission denied" for files in appdata, such as appdata/keystore.jks.	appdata does not belong to the container's user ID. b1 does not exit: it keeps running without being able to start, and WebAdmin waits for it.	On a Linux host, run <code>sudo chown -R 1000:1000 appdata</code> (65532:65532 for a hardened image), then run <code>docker compose restart b1</code> and <code>docker compose up -d</code> . On Docker Desktop the fix is the opposite: <code>chown</code> is what locks BridgeLink out of appdata, so do not run it, unless your project folder is inside a WSL2 Linux filesystem, where files keep their real ownership.
The command <code>docker compose up -d</code> ends with "dependency failed to start: container bridgelink-bl-1 is unhealthy", but only after about six minutes	b1 never became healthy, so WebAdmin was not started. Docker marks b1 unhealthy only after it has had about six minutes to start.	Run <code>docker compose logs b1</code> and read the last lines. A "Permission denied" error on appdata is a common cause, described in the row above. A database login failure is another, described in the row below.
The b1 container exits and restarts again and again, and <code>docker compose up -d</code> never finishes	BridgeLink cannot connect to or log in to the database. Each restart resets the health start period, so <code>docker compose up -d</code> keeps waiting. A common cause is changing POSTGRES_PASSWORD after the pgdata volume was first created. With the standard image, a `	in MP_DATABASE_PASSWORD` leaves the database password empty, with the same result. SQL Server only: its certificate is self-signed, comes from an internal certificate authority, or does not match the host name in the URL, which BridgeLink 26.9 does not accept by default.

Symptom	Likely cause	What to do
The first <code>docker</code> command fails with "permission denied while trying to connect to the docker API at <code>unix:///var/run/docker.sock</code> "	Your account is not allowed to use Docker	Add your account to the docker group with <code>sudo usermod -aG docker \$USER</code> , then sign out and back in, or put <code>sudo</code> in front of every docker command. See Run Docker Without Root .
The <code>b1</code> container starts, but port 8443 does not answer after the container was recreated. The log shows "Running" but never "Web server running at", and it mentions the keystore.	The keystore passwords changed, so BridgeLink cannot open the keystore in <code>appdata</code>	Set <code>MP_KEYSTORE_STOREPASS</code> and <code>MP_KEYSTORE_KEYPASS</code> back to the values used when <code>appdata/keystore.jks</code> was created.
The <code>webadmin</code> container exits at once or keeps restarting	The license terms were not accepted: <code>.env</code> is missing, is not in the folder with <code>docker-compose.yml</code> , or its line is not exactly <code>BL_ACCEPT_LICENSE=1</code>	Fix <code>.env</code> and run <code>docker compose up -d webadmin</code> .
WebAdmin sign-in shows "Could not reach the BridgeLink server. Verify the server address is correct and the server is running."	<code>b1</code> is not running or not healthy yet, or <code>BRIDGELINK_SERVER_URL</code> is wrong	Check <code>docker compose ps -a</code> , and confirm the URL is <code>https://b1:8443</code> .
The browser warns about the certificate	WebAdmin still uses its self-signed certificate	Click through the warning, or replace the certificate. See Replace the WebAdmin Certificate . If the browser offers no way to continue, it is enforcing HSTS for that name. See When a browser stops letting you click through under Sign In for the First Time , and To Undo the Swap in Replacing the Certificates to clear it.

For more WebAdmin problems, see [Troubleshooting](#) in the **BridgeLink WebAdmin Setup Guide**.

Support

Customers with an active Innovar Healthcare support plan can contact support at support@innovarhealthcare.com. Include the BridgeLink image and tag (or digest), your Compose file with the passwords removed, what you were doing, and the most recent part of `docker compose logs b1`. `docker compose logs b1` can contain message content, which may include patient data. Remove patient data before you send a log, or ask support for a secure way to send it.