



BiDiB-Broker

BiDiB-Broker ist eine Java Anwendung für das BiDiB-System zur Kommunikation mit BiDiBus und netBiDiB-Klienten via TCP/IP. Die Implementierung folgt der Protokollerweiterung [netBiDiB](#) des universellen [BiDiB](#)-Protokolls.



Der BiDiB-Broker (im Folgenden: **Broker**) dient als Mittler zum BiDiBus (RS232) und zwischen TCP/IP-fähigen Anwendungen und Geräten. (Stichwort: **BiDiB goes Netzwerk**)

An Brokers Seite stehen kleinere „Familienmitglieder“ mit einer so genannten REST-Schnittstelle zur Bedienung via Browser oder Apps.

Die Hauptanwendungsmöglichkeiten Brokers:



Adapter zwischen BiDiBus (USB, RS232) und TCP/IP



Hub für BiDiBus sowie netBiDiB-Anwendungen und -Geräte



*Standalone Buskopf für netBiDiB-Bediengeräte ^{*1)}*

In allen Fällen kann die Ethernet-Netzverbindung mit LAN oder WLAN erfolgen.

Broker spielt seine größten Vorteile auf einem Einplatinencomputer aus wie Raspberry Pi oder Banana Pi. Über eine LAN- oder WLAN-Verbindung lassen sich die verschiedenen Einstellungen mit Konfigurationsgeräten wie dem BiDiB-Monitor bzw. -Wizard komfortabel durchführen. Aber auch auf dem PC ist Broker universell einsetzbar.

Der direkte Link zur [aktuellen Broker-Version](#). Ein Link auf die jeweils aktuellste Broker-Version befindet sich auch im Anwenderhandbuch.

Zur Zeit liegen die folgenden erläuternden Handbücher vor:



[Anwenderhandbuch](#) - Grundsätzliche Erklärungen zu Inbetriebnahme und Betrieb



[Entwicklerhandbuch](#) - Hintergrundinformationen mit Konfigurationsmöglichkeiten



[Handbuch Embedded Systems](#) - Beschreibung Inbetriebnahme Raspberry Pi inkl. Aufbau Pairing-LED und -Taster



siehe auch [Fichtelbahn Newsletter 71, Seite 7ff v. Dez/22](#)

^{1*)} In der Entwicklung (Stand 01.02.2023) - Fragen nach dem Auslieferungstermin sind zwecklos



...

BiDiB-Broker-Family

Die Familienmitglieder als kleine Helferlein:

- [BiDiB-Registrierung](#) zum unkomplizierten Pairing
- [BiDiB-Harvester](#) zum „Einsammeln“ eines Knotenbaums
- [BiDiB-Barker](#) zum Auslesen einer Knotenkonfiguration

Das Familienmitglied für die Verbindung mit dem „Z21 LAN Protokoll“ © von Roco :

- [BiDiB-Z21Lan-Agent](#) (in Arbeit)

FAQ



Zusammenfassung der Fragen aus dem Forum

Wie kann ich Broker mit der Desktop-Verknüpfung in Windows starten?

Laut Handbuch wird empfohlen, für den vereinfachten Start Brokers eine passende Batchdatei von der Fichtelbahnseite in das Broker-Verzeichnis zu laden und anschließend eine Verknüpfung auf dem Desktop abzulegen.

Wenn man die Verknüpfung beim ersten Mal aufruft, verweigert Windows u.U. den Start. Das lässt sich mit folgenden zwei Klicks beheben:

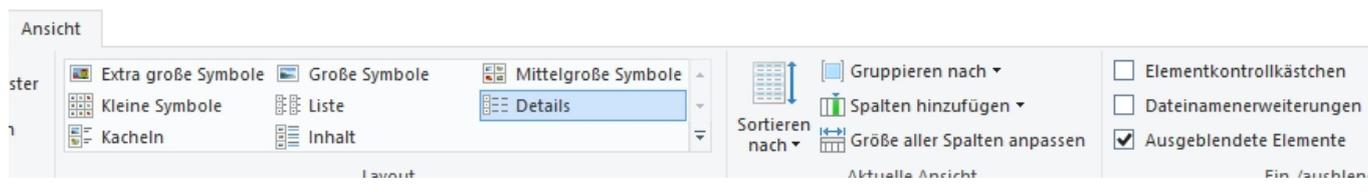


Wie finde ich die Konfigurations- und Protokolldateien?

Die Konfigurations- und Protokolldateien sind in einem Verzeichnis mit vorangestelltem Punkt gespeichert. Dadurch sind sie in der Standardeinstellung ausgeblendet, da sie in der Regel nicht vom Anwender benötigt werden.

Hier ein [Tipp vom Anwender "wissbegierig"](#), wie man diese Verzeichnisse unter Windows sichtbar machen kann:

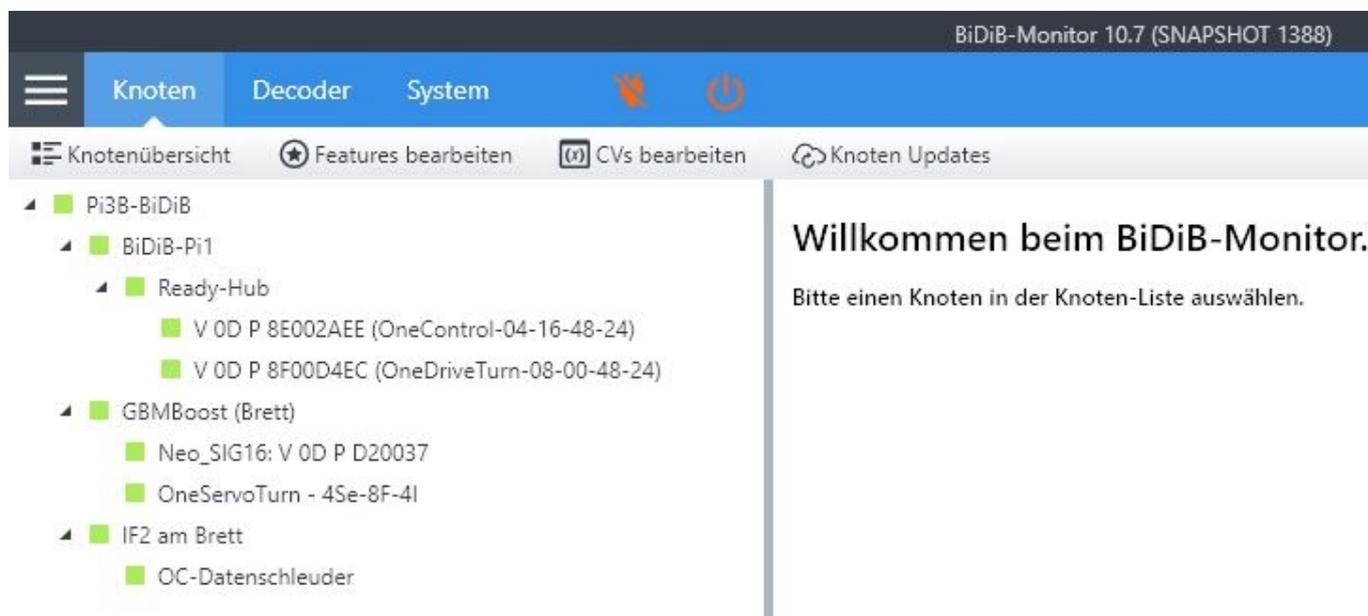
„Um dies zu sehen, ist in Windows ein Haken zu setzen: Ausgeblendete Elemente“



Kann ich mit dem BiDiB-Broker mehr als ein BiDiB-Interface bedienen?

Ja, das funktioniert mit Broker als Hub, also in der Standardeinstellung.

Eine Anwendungsmöglichkeit besteht darin, einen GBMBoost zum Fahren und einen weiteren zum Programmieren zu verwenden:



Im Beispiel sehen wir den BiDiB-Monitor, der über (W)LAN mit dem BiDiB-Broker (Pi3B-BiDiB) auf einem Raspberry Pi4b+ verbunden ist. Am Raspi sind angeschlossen ein BiDiB-Pi1 (ebenfalls mit einem Broker ausgestattet) über die serielle Schnittstelle, ein GBMBoost und ein IF2 über USB-Schnittstellen. An den drei BiDiB-Interface-Baugruppen sind unterschiedliche Knoten angeschlossen.

Was muss ich beachten, wenn ich mehr als ein BiDiB-Interface in meinem System verwenden möchte?

Auf die [Frage vom Anwender zrhbert](#):

„Kann ich den Bus in mehrere Teile zerlegen, so dass die einzeln per netBidibB angeschlossen sind? Einerseits scheint mir das möglich, z.B. bei zwei oder drei GMBBoost Mastern jeweils einen Avatar am USB-Interface vorschalten. Aber bekomme ich dann ein Problem, weil das DCC-Signal dann von mehreren unterschiedlichen Mastern erstellt wird?“

Bei einer normalen Anlage sind die Gleisabschnitte von verschiedenen Boostern immer wieder mal durch Fahrzeuge über der Trennstelle verbunden. Daher ist es zwingend erforderlich, nur eine DCC-Quelle zu haben.

Eine Aufteilung ist u.U. nur dann sinnvoll, wenn die DCC-Signal-Erzeuger unabhängige Bereiche versorgen, also z.B. Eisenbahn, Straßenbahn und Schmalspur-Bahn. Oder bei sehr großen Anlagen und **Einsatz besonderer Übergabeschaltungen** zwischen den DCC-Domänen.

Anders sieht es beim „Schalten und Walten“ aus. Da kann eine Aufteilung sinnvoll sein, wenn z.B. Steuerungsprogramme mit mehreren „Zentralen“ zusammenarbeiten können. Dafür würden sich auch statt eines GMBBoost das Gespann IF2+Raspi+Broker oder später IFnet alleine anbieten.

Das GMBBoost- bzw. IF2-Interface wird nicht gefunden. Was kann ich tun?

Zur [E-Mail-Anfrage von Renegee](#):

„Der BiDiB-Broker-Hub läuft auf einem PC mit Windows 11 Betriebssystem (java -jar bidib-broker-latest.jar). Das Pairing mit dem Wizard funktioniert, aber beim versuchten Verbindungsaufbau wird der angeschlossene GMBBoost nicht vom Broker-Hub gefunden. Die direkte Verbindung zwischen Wizard und GMBBoost funktioniert dagegen einwandfrei. Auf einem Raspi 4 zeigt sich das gleiche Bild: Wizard OK, Broker NOK.“

Aus unterschiedlichen Gründen kann das automatische Auffinden eines GMBBoost- oder IF2-Interfaces, also der USB-/seriellen Schnittstelle, im Broker misslingen. Das kann auch passieren, obwohl der Wizard die Verbindung anzeigt.

In diesem Fall kann man Broker mitteilen, die automatische Erkennung abzuschalten und stattdessen bestimmte Port(s) zu bevorzugen.

Dazu muss er den Aufruf erweitern z.B. zu:

Betriebssystem	Beispiel
Windows	java -jar bidib-broker-latest.jar --serial-device.preferred-ports=com3
Raspian	java -jar bidib-broker-latest.jar --serial-device.preferred-ports=/dev/serial0

Mehrere bevorzugte Ports können mit Komma getrennt (ohne Leerzeichen) angegeben werden.



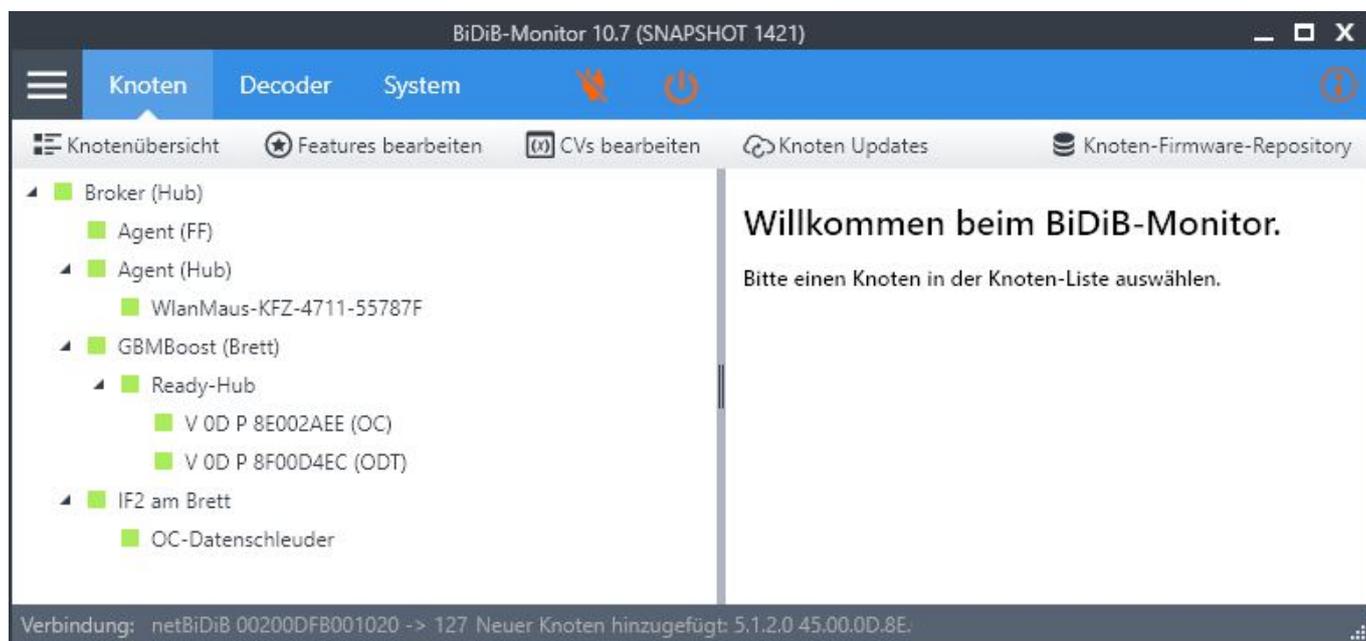


Die Bezeichnung der bevorzugten seriellen Schnittstelle kann bei Bedarf über den Wizard ermittelt werden.

Können sich mehrere netBiDiB-Knoten automatisch am BiDiB-Broker anmelden?

Ja, das ist eine der Kernaufgaben vom Broker und mit der Version 0.12.3-SNAPSHOT implementiert.

Das „Discovery“ (mDNS) ermöglicht eine automatische Verbindung ohne Eingriff des Anwenders.



Im Beispiel:

1. 1 x Z21Lan-Agent
2. 1 x Z21Lan-Agent (Hub), der eine WLANmaus bedient
3. 1 x GBMBoost mit Anhang
4. 1 x IF2 mit einer „Datenschleuder“ für ein wenig Traffic

Eine ausführliche Beschreibung von Wolfgang Kufer findet sich im Forum: [IFnet, wie findet man die IP?](#). Das Vorgehen gilt natürlich auch für den BiDiB-Broker.

Können netBiDiB-Knoten ohne automatische Anmeldung (mDNS) verbunden werden?

Ja, neben der automatischen Anmeldung können auch netBiDiB-Knoten, die die automatische Anmeldung (mDNS) nicht beherrschen, verbunden werden.

Dazu muss ein Anwender die Netzwerkadresse kennen und dem netBiDiB-Knoten mitteilen. Wir haben für diese Fälle einen „Gasteingang“ definiert. Broker hört am „Gasteingang“ auf Port 62874.



Netzwerkadresse und Port zeigt Broker übrigens auf seiner Pairing-GUI an.

Dieses Verfahren sollte allerdings nur in Ausnahmefällen Verwendung finden!

Siehe auch die ausführliche Beschreibung von Wolfgang Kufer im Forum: [IFnet, wie findet man die IP?](#). Das Vorgehen gilt natürlich auch für den BiDiB-Broker.

Wie kann ich Broker im Raspberry Pi automatisch starten lassen?

Es gibt mehrere Möglichkeiten. Eine davon ist, die Datei rc.local entsprechend zu ergänzen:

```
Edit File als „sudo“: /etc/rc.local (vor dem Ende (exit 0) einfügen, z.B.:  
cd /etc  
sudo nano rc.local  
◦ cd /home/pi/BiDiB/  
◦ sudo -u pi java -jar bidib-broker-latest.jar  
Anmerkung: Zusätzlich gewünschte Startparameter anhängen
```

Wie kann Broker starten, obwohl die gewünschten Ports belegt sind?

Es kann passieren, dass andere (BiDiB-)Anwendungen die gleichen Ports wie Broker verwenden. Das wird im Broker-Protokoll als Fehler angezeigt, z.B.:

```
2023-03-07 17:13:04.589 ERROR 3516 --- [ pool-6-thread-1 ] o.s.i.i.t.c.TcpNetServerConnectionFactory - Error on ServerSocket; port = 62874  
java.net.BindException: Address already in use: bind  
at java.base/sun.nio.ch.Net.bind0(Native Method)  
-----  
2023-03-07 17:13:04.605 ERROR 3516 --- [ pool-4-thread-1 ] o.s.i.i.t.c.TcpNetServerConnectionFactory - Error on ServerSocket; port = 62875  
java.net.BindException: Address already in use: bind  
at java.base/sun.nio.ch.Net.bind0(Native Method)  
at java.base/java.net.ServerSocket.<init>(ServerSocket.java:274)
```

Diese Ports stehen Broker für die netBiDiB-Verbindungen nicht mehr zur Verfügung. Aber Broker kann im Bedarfsfall mit jeweils anderen Ports gestartet werden.

Der Port 62874 - für „einfache“ netBiDiB-Knoten, die „Bonjour“ nicht beherrschen - ändert man mit z.B.:

```
java -jar bidib-broker-latest.jar --connection.side-entrance.port=62870
```

Allerdings muss man die Portnummer in allen „einfachen“ Anwendungen ebenfalls anpassen.

Der Port 62875 ist für die Kommunikation mit einem Host-System zuständig, wie z.B. dem Wizard. Änderung der Portnummer, z.B.:

```
java -jar bidib-broker-latest.jar --master-data.tcp-port-number=62877
```

Die Portnummer muss natürlich frei sein und auch im Host-System entsprechend angepasst werden.



Alle Startparameter können, getrennt durch Leerzeichen, hintereinander geschrieben werden.

Ich brauche den voreingestellten Port 62875 für andere Zwecke. Kann sich ein Host trotzdem mit dem Broker verbinden?

Ja, das funktioniert. Allerdings muss man Broker Bescheid geben, [wie auch oben beschrieben](#). Den Port für diesen „Haupteingang“ verschiebt man mit dem zusätzlichen Startparameter, z.B.:

```
java -jar bidib-broker-latest.jar --master-data.tcp-port-number=62900
```

Für den Host ist der Broker in diesem Beispiel jetzt über Port 62900 erreichbar.



Der Bereich für frei definierbare Ports liegt zwischen 49152 und 65535.

Kann ich die Anzeige der Meldungen in der Konsole irgendwie reduzieren? Also z.B. DEBUG ausblenden?

Ja, das kann man im laufenden Betrieb oder mit einem Startparameter einstellen.

Im laufenden Betrieb von der Kommandozeile aus (im Beispiel für 127.0.0.1=localhost):

```
curl -X "POST" "http://127.0.0.1:62876/actuator/loggers/org.bidib.broker" -H "Content-Type: application/json; charset=utf-8" --data "{\"configuredLevel\": \"INFO\"}"
```

Mit Startparameter - Version 1:

```
java -jar bidib-broker-latest.jar --logging.level.org.bidib.broker=INFO --logging.level.org.bidib.springbidib=INFO
```

Mit Startparameter - Version 2:

```
java -jar bidib-broker-latest.jar --logging.config=classpath:logback-spring-min-con.xml
```

Kann man beim Start von Broker mehrere Parameter angeben?

Ja, das geht natürlich. Oben ist jeweils nur der für das Beispiel wichtige Parameter aufgeführt.

Alle Parameter müssen dabei durch ein Leerzeichen voneinander getrennt und im Bedarfsfall mit „Gänsefüßchen“ umschlossen werden.

Die Startzeile für Broker mit oben aufgeführten Parameter würde wie folgt aussehen:

```
java -jar bidib-broker-latest.jar --connection.side-entrance.port=62873 --master-data.tcp-port-number=62877 --server.port=62879 --logging.level.org.bidib.broker=INFO
```

Was tun, wenn Broker mit der Meldung "Web server failed to start. Port 62876 was already in use." abbricht?

Unter widrigen Umständen kann der Start Brokers mit der folgenden Meldung scheitern:

```
Error starting ApplicationContext. To display the conditions report re-run your application with 'debug' enabled.
2023-03-07 16:29:52.020 ERROR 4788 --- [main] o.s.b.d.LoggingFailureAnalysisReporter -
*****
APPLICATION FAILED TO START
*****
Description:
Web server failed to start. Port 62876 was already in use.
Action:
Identify and stop the process that's listening on port 62876 or configure this application to listen on another port.
```

In diesem Falle verwendet eine andere Anwendung der Port 62876, das für die Pairing-GUI zuständig ist. Oder Broker ist irregulär beendet worden und konnte den Port nicht mehr schließen. Damit steht der Port für den aktuellen Broker nicht mehr zur Verfügung.

Wenn man die Anwendung kennt, sollte sie beendet und nach Broker gestartet werden. Hat jetzt die andere Anwendung das Problem, bleibt die Möglichkeit Broker mit einem anderen Port zu starten, z.B.:

```
java -jar bidib-broker-latest.jar --server.port=62879
```

In diesem Beispiel muss der Port ausprobiert werden und in der Browser-GUI entsprechend übernommen werden.



Ist die blockierende Anwendung nicht zu ermittelt, bleibt als Abhilfe nur die Brechstange, indem der Verursacher identifiziert und radikal beendet wird. Hier unter Windows am Beispiel Port 62876:

```
C:\BiDiB>netstat -ano | find "62876"
TCP 0.0.0.0:62876 0.0.0.0:0 ABHÖREN 11864
TCP [::]:62876 [::]:0 ABHÖREN 11864
```

```
C:\BiDiB>taskkill /pid 11864 /f
ERFOLGREICH: Der Prozess mit PID 11864 wurde beendet.
C:\BiDiB>
```



Auf dem Raspberry Pi reicht:

```
pi@Pi3B-BiDiB:~ $ sudo pkill -2 -f bidib-broker-latest.jar
```

Vorher und nachher prüfen, ob Broker läuft:

```
pi@Pi3B-BiDiB:~ $ ps ax | grep bidib-broker-latest.jar
```



Das Überschreiben der Datei **bidib-broker-latest.jar** im laufenden Betrieb und anschließendem Neustart wird dagegen nicht empfohlen, da man dem Prozess die Basis raubt - mit u.U. unvorhergesehene Nebenwirkungen!

From:

<https://forum.opendcc.de/wiki/> - **BiDiB Wiki**

Permanent link:

https://forum.opendcc.de/wiki/doku.php?id=bidib_broker_family

Last update: **2023/12/18 18:33**

