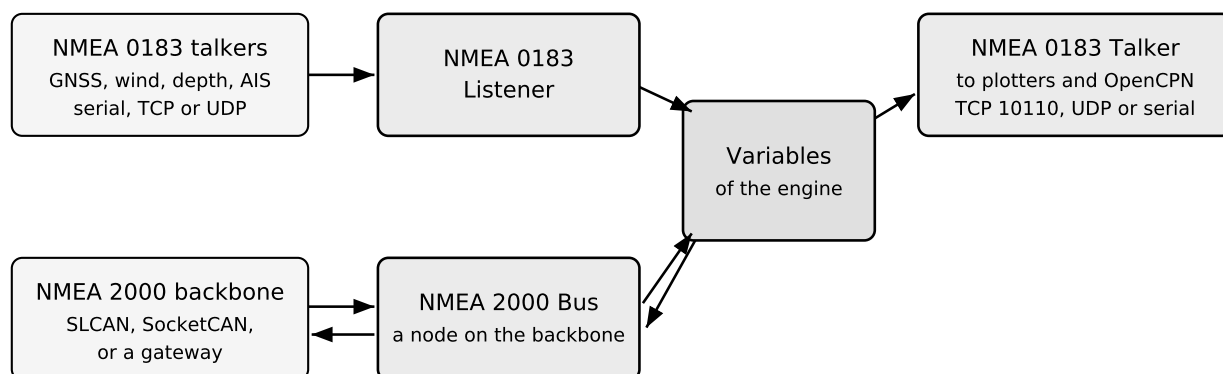


NMEA

Generated on October 14, 2026



NMEA is how a boat's instruments talk: a GNSS receiver's position, a wind instrument, a depth sounder, an AIS receiver, an engine. Three modules speak it. Two speak **NMEA 0183**, the line-of-text protocol where one talker drives a wire and anyone listens; one speaks **NMEA 2000**, the CAN backbone newer instruments share.

Module	What it does	Typical use
NMEA-0183-Listener (NMEA 0183 Listener)	Takes the fields of the sentences a talker sends into variables	A GNSS receiver's fix, a wind instrument, AIS targets, a multiplexer's feed
NMEA-0183-Talker (NMEA 0183 Talker)	Builds sentences from variables and serves them; takes what a client sends back into variables	Feeding a chart plotter or OpenCPN with values the instance holds
NMEA-2000 (NMEA 2000 Bus)	Takes parameter groups off a backbone into variables, and sends variables onto it as a node	Reading a vessel's position, depth, wind and engines without a gateway converting them

Enable, **Name** and the **Status and messages** tab work the same way in every module. They are described in [Settings every module shares](#).

Important

Data Orchestrator reads and writes NMEA 2000 parameter groups; it is not NMEA 2000 certified. Its definitions of the groups are canboat's, reverse-engineered from public sources and published under the Apache License 2.0, because the official database is for members only. The engine claims an address and answers what other devices ask a node, but holds no NMEA certification and no NMEA manufacturer code. Where a tender asks for certified equipment, put a certified gateway between the backbone and the engine.

Addresses

Every row of the three modules names a variable and an **address**: which value of which sentence or group it is.

NMEA 0183 addresses are `[talker]SENTENCE[selectors]:field`.

- `GGA:latitude` is the latitude of any talker's GGA sentence; `GPGGA:latitude` only that of the talker `GP`.
- A field's name is marineapi's, which the editor lists for each sentence: `GGA:satelliteCount`, `RMC:speed`, `MWV:angle`. A position is two fields, `latitude` and `longitude`; a time and a date are ISO 8601 text.
- AIS messages are addressed under `AIS`, whichever type they are and whether VDM or VDO carried them: `AIS[MMSI=257123456]:speedOverGround` is that vessel's speed from whichever position report last gave it.

NMEA 2000 addresses are `PGN[selectors]:field`.

- `128267:depth` is the depth of any sounder; `129025[src=12]:latitude` only the position from the device at source address 12, when two receivers are on the backbone.
- A field's name is canboat's, which the editor lists for each group with its unit. A field of a repeating set takes its position after a dot: `129540:snr.3` is the third satellite's.

A **selector** is a field that must hold a value for the sentence or group to count: `MWV[true=false]:angle` is the apparent wind angle and not the true one, which MWV sends under the same name, and `127488[instance=1]:speed` is the starboard engine's speed. In an NMEA 2000 address `src` is the sender and `dst` the destination; a lookup field may be selected by its name or its number. In a forwarder a selector is a field **sent** as a constant: `GPRMC[status=ACTIVE]:latitude` sends RMC marked valid. Names are matched whatever their case.

Units. NMEA 0183 values are in the sentence's own units - knots, degrees, metres - as the talker sent them. NMEA 2000 values are in SI units, as canboat defines each field: radians, metres per second, kelvin, pascal. A lookup field, such as a wind reference, holds its name as text (`Apparent`); a value the sender marks as not available is not written at all.

NMEA 0183 Listener

The `NMEA-0183-Listener` module listens to one talker, or to a multiplexer that combines several. **Collectors** take fields of its sentences into variables. NMEA 0183 is simplex: nobody answers a talker, so this module only listens.

Settings

Key	Setting	Type	Default	Meaning
<code>transport</code>	Transport	text	<code>serial</code>	<code>serial</code> reads a serial port, <code>tcp</code> connects to a multiplexer's feed, <code>udp</code> listens for broadcast sentences
<code>serialPort</code>	Serial port	text	(empty)	<code>serial</code> : the device, such as <code>/dev/ttyUSB0</code> or <code>COM3</code> . The editor offers the ports it finds on the server
<code>bauds</code>	Bauds	int	<code>4800</code>	<code>serial</code> : the line speed. NMEA 0183 runs at 4800; AIS receivers and fast multiplexers at 38400
<code>host</code>	Host	text	(empty)	<code>tcp</code> : the multiplexer's address
<code>port</code>	Port	int	<code>10110</code>	<code>tcp</code> : the port to connect to. <code>udp</code> : the port to listen on
<code>collectors</code>	Collector	list of rows	empty	The fields taken into variables

Collectors

Key	Column	Type	Default	Meaning
<code>field</code>	Variable	variable code	(none)	The variable the field is written to
<code>address</code>	Address	text	(empty)	The field, as Addresses says
<code>minIntervalMs</code>	Minimum interval [ms]	int	(empty)	The least time between two writes of the variable. Empty or 0: every change

The **Type / problem** column is not stored: it shows the variable type the field yields, or what is wrong with the address. The picker above the table lists every sentence the library reads and the fields of the one chosen, with their types; **Add as collector** adds a row for them.

Behaviour

- **Opening.** The transport, and every row's address, are checked first: an address that does not parse, or names a sentence or field the library does not have, stops the module opening, with a message naming the row.
- **The source.** On `serial` and `tcp` the module keeps the port or the connection open, and opens it again every 2 seconds when it goes. On `udp` it listens on the port, on every network interface.
- **Only what is mapped is read.** A sentence no collector names is passed over without being parsed.
- **Checksums.** A sentence that carries a checksum must match it; one that does not is dropped. A sentence with no checksum is taken as it is. Anything before the `$` or `!` - a tag block, a gateway's timestamp - is ignored.
- **Lost sentences.** A sentence with a bad checksum, or one the library cannot read, is dropped and counted. While sentences keep being lost the module warns, with the count; the warning ends once a minute has passed without one. A bad sentence costs that sentence and nothing else.
- **AIS.** A message split over several VDM sentences is put back together before it is read. A message type the library does not decode is passed over.
- **Values.** A field the talker left empty writes nothing: an empty field is the talker saying it does not know. A value lands only when it differs from the last one written, and no sooner than `minIntervalMs` after it; the last value is never lost, it waits for the interval to end. Values are written through the engine's event queue.

Messages

All texts are in English; the full list is in [Module messages](#). "then `ERROR`" follows `errorAfterSeconds` in [Settings every module shares](#). `{server}` and `{bus}` name the source as `NMEA 0183 on /dev/ttyUSB0` or `NMEA 0183 at host:port`; `{mapping}` is the variable's code.

When	Level	Text
Opening: a transport that is not one of the three	ERROR	'{transport}' is not serial, tcp or udp
Opening: <code>serial</code> with no port, or <code>tcp</code> with no host	ERROR	No serial port or No host
Opening: an address that does not parse	ERROR	{mapping}: {reason}
Opening: a sentence the library does not read	ERROR	{mapping}: {sentence} is not a sentence this module reads
Opening: a field the sentence does not have	ERROR	{mapping}: {sentence} has no field '{field}'
Opening, <code>udp</code> : the port cannot be bound	ERROR	Cannot listen on port {port}: {reason}
<code>udp</code> : listening	INFO	Listening on port {port}
The port or connection is open	INFO	Connected to {server}
The port or connection could not be opened, or was lost	WARNING, then ERROR	Disconnected from {server}, then Bus {bus} unavailable: {reason}
Sentences with a bad checksum	WARNING	Bad checksum on {n} sentence(s)
Sentences the library could not read	WARNING	{n} unreadable sentence(s) discarded: {reason}
A value its variable's declared type refuses	WARNING	Output {output} refused {value}: {reason}
The engine's event queue is full	WARNING	Event queue full: {count} events pending, {dropped} dropped

Tip

On Linux, the account the service runs as needs access to the serial device, usually through the group that owns it (`dialout` on Debian and Ubuntu). Without it the module reports the port as unavailable.

Example

A GNSS receiver on a USB serial adapter, and the AIS position of one vessel from the same multiplexed line:

```

Transport      serial
Serial port    /dev/ttyUSB0
Bauds          38400

Collector      GNSS_LAT      GGA:latitude
Collector      GNSS_LON      GGA:longitude
Collector      GNSS_SATS     GGA:satelliteCount
Collector      COG           RMC:course           1000
Collector      TUG_SOG       AIS[MMSI=257123456]:speedOverGround 5000

```

The position follows every GGA. The course is written at most once a second, and the tug's speed at most every five seconds, whatever rate the talker sends at.

NMEA 0183 Talker

The `NMEA-0183-Talker` module builds NMEA 0183 sentences from variables and sends them. Served on TCP it is a server, as the Modbus TCP slave is: chart plotters, OpenCPN and other engines connect to its port, port 10110 by convention, and each is served the same sentences. It can also broadcast on UDP, or be the talker on a serial line.

Forwarders build the sentences. **Collectors** take the sentences a client sends back on its connection, or the serial port receives - an autopilot's heading, a route - into variables, exactly as the listener's do.

Settings

Key	Setting	Type	Default	Meaning
<code>transport</code>	Transport	text	<code>tcp-server</code>	<code>tcp-server</code> serves clients, <code>udp</code> broadcasts, <code>serial</code> drives a serial port
<code>port</code>	Port	int	<code>10110</code>	<code>tcp-server</code> : the port to listen on, on every network interface. <code>udp</code> : the port sent to
<code>maxClients</code>	Maximum clients	int	<code>8</code>	<code>tcp-server</code> : how many clients are served at once. One more is refused
<code>host</code>	Host	text	<code>255.255.255.255</code>	<code>udp</code> : where to send. The default is the local network's broadcast address
<code>serialPort</code>	Serial port	text	(empty)	<code>serial</code> : the device
<code>bauds</code>	Bauds	int	<code>4800</code>	<code>serial</code> : the line speed
<code>talkerId</code>	Talker identifier	text	<code>II</code>	The talker of a sentence whose address names none. <code>II</code> is integrated instrumentation
<code>forwarders</code>	Forwarder	list of rows	empty	The fields sent from variables. The stored key is spelled <code>forwarders</code>
<code>collectors</code>	Collector	list of rows	empty	The fields taken from what clients send back

Forwarders and collectors

A forwarder row uses `field`, `address` and `intervalsMs`. A collector row uses `field`, `address` and `minIntervalsMs`.

Key	Column	Type	Default	Meaning
<code>field</code>	Variable	variable code	(none)	Forwarder: the variable a field is built from. Collector: the variable written
<code>address</code>	Address	text	(empty)	The field, as Addresses says
<code>intervalsMs</code>	Interval [ms]	int	(empty)	Forwarders: also send the sentence on this clock, whether or not anything changed. Empty or 0: on change only
<code>minIntervalsMs</code>	Minimum interval [ms]	int	(empty)	Collectors: the least time between two writes of the variable

Rows of one sentence make one sentence. Every forwarder whose address names the same talker, sentence and selectors fills the same sentence: `GPRMC:latitude`, `GPRMC:longitude` and `GPRMC:speed` are one RMC line. It is sent when any of their variables changes, with the current value of each, and on the shortest interval any of them asks for. The picker is the sentence builder: it lists the sentences the library can write and, for the one chosen, the fields it can fill.

Behaviour

- **Opening.** Every forwarder is checked: its sentence and field must be ones the library can write, a position needs both `latitude` and `longitude` in the same sentence, AIS is refused because it is received and never sent, and the talker must be a known one. Collectors are checked as the listener's are. `tcp-server` binds its port here, so a port another program holds is the module's status at Start.
- **Building.** `marineapi` encodes the sentence and its checksum. A variable that holds nothing yet leaves its field empty, which is what NMEA 0183 means by unknown. A selector is a field sent as a constant.
- **Clients.** Each client is served from a queue of its own, of 256 sentences. A client that stops reading is dropped when its queue fills, so it never slows the others; one that disconnects takes nothing with it.
- **What clients send back.** Each client's lines are read as the listener reads a talker's, checksums and counting included, and land in the collectors' variables.
- **Plotters.** A chart plotter plots RMC only when it is marked valid: send it as `RMC[status=ACTIVE]:...`.

Messages

All texts are in English; the full list is in [Module messages](#). `{message}` is the sentence, as `GPRMC[status=ACTIVE]`.

When	Level	Text
Opening: a transport that is not one of the three	ERROR	'{transport}' is not tcp-server, udp or serial
Opening: <code>serial</code> with no port, or <code>udp</code> with no host	ERROR	No serial port or No host
Opening, <code>udp</code> : the host cannot be resolved	ERROR	Cannot send to {host}: {reason}
Opening: an address that does not parse, or names nothing	ERROR	{mapping}: {reason}, {mapping}: {sentence} is not a sentence this module reads or {mapping}: {sentence} has no field '{field}'
Opening: AIS in a forwarder	ERROR	{mapping}: AIS is received, not sent
Opening: a field the library cannot write	ERROR	{mapping}: {field} cannot be written into {sentence}
Opening: half a position	ERROR	{mapping}: a position needs both latitude and longitude
Opening: an unknown talker	ERROR	{mapping}: '{talker}' is not a talker identifier
Opening, <code>tcp-server</code> : the port cannot be bound	ERROR	Cannot listen on port {port}: {reason}
<code>tcp-server</code> : listening	INFO	Listening on port {port}
A client connected	INFO	Client {address} connected
A client disconnected	WARNING	Client {address} disconnected, cleared when it connects again
One client more than <code>maxClients</code>	WARNING	Invalid request discarded: {address} would be client {n} of {max}
A client dropped for not reading	WARNING	Client {address} dropped: it did not keep up
<code>serial</code> : the port opened, or was lost	INFO, or WARNING then ERROR	Connected to {server}, or Disconnected from {server} then Bus {bus} unavailable: {reason}
A sentence could not be built or sent	WARNING	Transmitting {message} failed: {reason}
Sentences from clients with a bad checksum, or unreadable	WARNING	Bad checksum on {n} sentence(s) or {n} unreadable sentence(s) discarded: {reason}
A value its variable's declared type refuses, or the event queue is full	WARNING	Output {output} refused {value}: {reason} or Event queue full: {count} events pending, {dropped} dropped

Example

The bridge feed: a position and speed every second, apparent wind every half second, and the heading an autopilot sends back:

```

Transport      tcp-server
Port           10110
Maximum clients 8
Talker identifier GP

Forwarder      GNSS_LAT      RMC[status=ACTIVE]:latitude      1000
Forwarder      GNSS_LON      RMC[status=ACTIVE]:longitude
Forwarder      SOG           RMC[status=ACTIVE]:speed
Forwarder      WIND_ANGLE    WIMWV[true=false]:angle          500
Forwarder      WIND_SPEED    WIMWV[true=false]:speed
Collector      AUTOPILOT_HDG  HDG:heading

```

OpenCPN connected to port 10110 plots the position from `GNSS_LAT` and `GNSS_LON`, and a client that writes an HDG sentence back on its connection sets `AUTOPILOT_HDG`.

NMEA 2000 Bus

The `NMEA-2000` module joins an NMEA 2000 backbone. **Collectors** take fields of the parameter groups other devices send into variables. **Forwarders** send variables onto the backbone, as parameter groups the module sends as a node.

NMEA 2000 is J1939 over CAN at 250 kbit/s, so the module reaches the backbone the way the [CAN modules](#) reach a bus, and shares their transport layer. A long group travels as a fast packet, or through the J1939 transport protocol; the module puts both back together and splits what it sends.

Settings

Key	Setting	Type	Default	Meaning
<code>transport</code>	Transport	text	<code>slcan</code>	<code>slcan</code> , <code>socketcan</code> , <code>yd-serial</code> , <code>yd-tcp</code> , <code>actisense</code> or <code>loopback</code> , as Reaching the backbone says
<code>serialPort</code>	Serial port	text	(empty)	<code>slcan</code> , <code>yd-serial</code> , <code>actisense</code> : the device
<code>bauds</code>	Bauds	int	<code>115200</code>	The serial speed of the adapter or gateway
<code>interface</code>	Interface	text	<code>can0</code>	<code>socketcan</code> : the Linux interface. <code>loopback</code> : the loopback's name
<code>host</code>	Host	text	(empty)	<code>yd-tcp</code> : the gateway's address
<code>port</code>	Port	int	<code>1457</code>	<code>yd-tcp</code> : the port the gateway serves its RAW stream on
<code>sourceAddress</code>	Source address	int	<code>100</code>	The address the module claims first, 0 to 253. Taken, it claims the next free one
<code>deviceInstance</code>	Device instance	int	<code>0</code>	The device instance in the module's NAME, lower three bits and upper five
<code>systemInstance</code>	System instance	int	<code>0</code>	The system instance in its NAME, 0 to 15
<code>collectors</code>	Collector	list of rows	empty	The fields taken into variables
<code>forwarders</code>	Forwarder	list of rows	empty	The fields sent from variables. The stored key is spelled <code>forwarders</code>

Collectors and forwarders

A collector row uses `field`, `address` and `minIntervalMs`. A forwarder row uses `field`, `address` and `intervalMs`.

Key	Column	Type	Default	Meaning
<code>field</code>	Variable	variable code	(none)	Collector: the variable written. Forwarder: the variable a field is sent from
<code>address</code>	Address	text	(empty)	The field, as Addresses says
<code>minIntervalMs</code>	Minimum interval [ms]	int	(empty)	Collectors: the least time between two writes of the variable
<code>intervalMs</code>	Interval [ms]	int	(empty)	Forwarders: also send the group on this clock. Empty or 0: on change only

Every forwarder whose address names the same group, selectors and destination fills the same group, which is sent when any of their variables changes and on the shortest interval any of them asks for. A field the forwarders do not give is sent as not available. Only the fields before a repeating set can be sent. The picker above the tables lists every group canboat defines and the fields of the one chosen, with their units.

Reaching the backbone

Transport	Reaches the backbone through	The module is
<code>slcan</code>	An SLCAN USB adapter (CANable, USBtin) on a serial port, set to 250 kbit/s by the module	A node: it claims an address
<code>socketcan</code>	A Linux SocketCAN interface, whose bit rate the system sets. Linux only	A node
<code>yd-serial</code> , <code>yd-tcp</code>	A Yacht Devices gateway in RAW mode, over its USB port or its network port	A node
<code>actisense</code>	An Actisense NGT-1, which hands over whole groups	Not a node: the gateway is, and sends for it
<code>loopback</code>	A bus inside the engine, for tests and demonstrations	A node

Joining as a node

A device may not send on an NMEA 2000 backbone before it has **claimed an address**. Through every transport but `actisense`, the module claims `sourceAddress` when the bus comes up, waits 250 ms for another device to contest it, and moves to the next free address if one with a better NAME does. A module that finds no free address says so and sends nothing; it reclaims whenever the bus comes back. Two modules, or two engines, never claim with the same NAME: its unique number comes from the installation and the module's name.

Its NAME says what it is: a PC gateway among internetwork devices, in the marine industry group. It carries no manufacturer code, because Amtiri holds none.

A node must also answer what other devices ask it, or a plotter may not list it:

Request	Answer
ISO Request (59904) for the address claim (60928)	The module's claim
ISO Request for product information (126996)	Model <code>DataOrchester</code> , the engine's version, the canboat release, the installation's identifier. Certification level is left unknown
ISO Request for the PGN lists (126464)	The groups it sends - its forwarders' and the administration groups - and the groups it receives
ISO Request for a group one of its forwarders sends	That group, now
ISO Request addressed to it for anything else	An ISO Acknowledgement (59392), NAK
Group function (126208), Request	As an ISO Request, acknowledged with "PGN not supported" when the module does not have it
Group function, Command to the address claim	Fields 3 and 4 set the device instance and field 8 the system instance; the module acknowledges and claims again with its new NAME. Any other command is acknowledged with "PGN not supported"

Instances set from the backbone last until the engine restarts; the ones in the module's settings are the lasting ones. Behind an Actisense gateway, the gateway is the node: it claims its own address, answers for itself, and the module's forwarders are sent through it.

Behaviour

- **Opening.** The transport, the source address and every row are checked first: an address that does not parse, a group canboat does not define, a field the group does not have, a repeated field without its position, a repeated field in a forwarder, or `src` in a forwarder stop the module opening, with a message naming the row. A missing adapter does not: the module opens, says the bus is unavailable, and picks the adapter up when it appears.
- **Receiving.** Fast packets and J1939 transport messages are put back together; a message whose frames stop arriving is dropped after 750 ms. Only the groups a collector names are decoded. A group with several definitions in canboat - a manufacturer's proprietary group - is read by the one whose match fields it carries.
- **Values.** A field the sender marks as not available writes nothing. A value lands only when it differs from the last one written, and no sooner than `minIntervalMs` after it, through the engine's event queue.
- **Sending.** A group goes out only while the module holds an address. Behind `actisense` , the gateway sends it.

Messages

All texts are in English; the full list is in [Module messages](#). `{bus}` and `{server}` name the transport, as `SLCAN on /dev /ttyACM0` or `Yacht Devices gateway at host:port` ; `{message}` is the group, as `127488[instance=0]` .

When	Level	Text
Opening: a setting that cannot work	ERROR	Configuration invalid: {reason}: '{transport}' is not an NMEA 2000 transport, No serial port, No interface, No host, A source address is 0 to 253, not {n}, SocketCAN is available on Linux only, not on {os}, or {bus} is already open at {bitrate} bit/s
Opening: an address that does not parse	ERROR	{mapping}: {reason}
Opening: a group canboat does not define	ERROR	{mapping}: {pgn} is not a parameter group the catalogue defines
Opening: a field the group does not have	ERROR	{mapping}: {pgn} ({name}) has no field '{field}'
Opening: a repeated field without its position	ERROR	{mapping}: {field} repeats, so it needs its position, {field}.1
Opening: a repeated field in a forwarder	ERROR	{mapping}: {field} repeats, and only the fields before a repeating set are sent
Opening: src in a forwarder	ERROR	{mapping}: src selects a sender, and this module sends as itself
The bus is up	INFO	Connected to {server}
The address is claimed	INFO	Address {address} claimed
The adapter or gateway could not be opened, or was lost	WARNING, then ERROR	Disconnected from {server}, then Bus {bus} unavailable: {reason}
The controller is error passive	WARNING	{bus} is error passive
The controller is bus-off	ERROR	{bus} is bus-off
Every address is taken	ERROR	No address could be claimed, so nothing is transmitted
A group could not be built or sent	WARNING	Transmitting {message} failed: {reason}
A value its variable's declared type refuses, or the event queue is full	WARNING	Output {output} refused {value}: {reason} or Event queue full: {count} events pending, {dropped} dropped

Example

A backbone reached through an SLCAN adapter: the position, the depth, the apparent wind and both engines, and a generator's speed sent to the backbone every 100 ms:

Transport	slcan		
Serial port	/dev/ttyACM0		
Source address	100		
Collector	LAT	129025:latitude	
Collector	LON	129025:longitude	
Collector	DEPTH	128267:depth	1000
Collector	AWA	130306[reference=Apparent]:windAngle	
Collector	RPM_PORT	127488[instance=0]:speed	500
Collector	RPM_STBD	127488[instance=1]:speed	500
Forwarder	GEN_RPM	127488[instance=2]:speed	100

`AWA` is in radians and `DEPTH` in metres. `GEN_RPM` goes out as engine instance 2 once the module has claimed its address, every 100 ms and whenever it changes.

Next steps

- [CAN](#)
- [Module catalogue](#)
- [Settings every module shares](#)
- [Module messages](#)
- [Connectivity](#)

This page describes Data Orchester engine 6.20.0.