

The Hermes Standard
for "M-to-M" in SMT Assembly

The Hermes Standard

The Hermes Standard Change Proposal

Foundation of vertical channel

Voting meeting:

28th of January 2019 (APEX / San Diego)

Requesting company:

Workgroup "Hermes Vertical"



The Hermes Standard for vendor independent machine-to-machine communication in SMT Assembly.

Version change:

Minor

Affected versions:

1.1

Service description tag:

FeatureConfiguration

Description:

~~Empowered~~ Enables the remote configuration channel to support a full extendable vertical channel for connections of a supervisory system.

Use cases:

Allow to establish a vertical channel for connection of a supervisory system.

Functionality / communication sequences:

See proposed standard changes.

New / changed XML messages:

See proposed standard changes.

Proposed changes to standard:

2 Technical concept

2.1 Prerequisites ~~and topology~~

This specification is based on the prerequisite, that any application implementing this protocol has to provide connectivity based on Internet Protocol (IP) [IETF_RFC_791]/[IETF_FRC_2460] via Transmission Control Protocol (TCP) [IETF_RFC_793] (ISO/OSI model [ISO_7498-1] layer 3) to the adjacent machines ~~and for communication with supervisory systems.~~

2.3 Machine-to-machine communication (horizontal channel)

2.3.1 Topology

Any machine in a line offers one TCP server per lane on its downstream side. Further servers per lane might also be necessary, e.g. if reverse transportation is supported. The TCP port number is not specified but can be configured by the user. The recommended port numbers are 50100 plus lane identifier (ID) with lanes being enumerated looking downstream from right to left beginning with 1 (e.g. for the left lane of a dual lane machine, the upstream machine server accepts connections on port 50102). For every further server plus 10 is recommended to be added to the port number.

The downstream machine opens one connection for every lane and every transportation interface on its upstream side to the upstream machine(s). So every PCB handover point corresponds to one TCP connection per exchange direction.

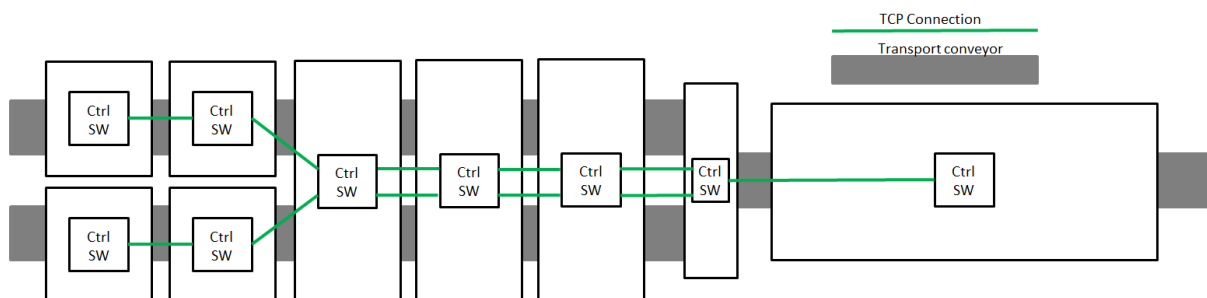


Fig. 2 TCP connections in a line

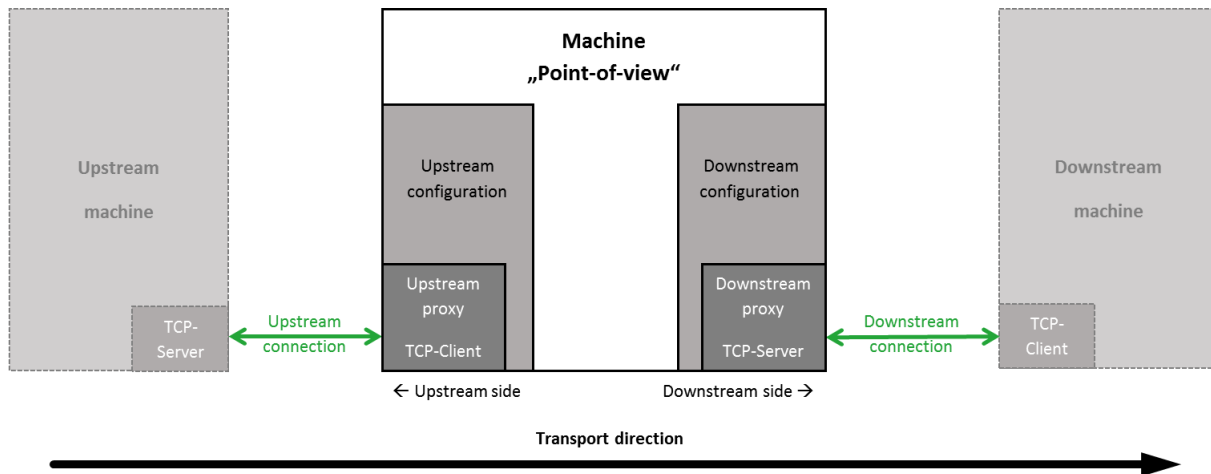


Fig. 3 Upstream and downstream from the perspective of the machine

2.3.2 Connecting, handshake and detection of connection loss

After booting, the downstream machine starts cyclic connection attempts to the configured upstream machines. When a connection is established, the downstream machine starts sending a ServiceDescription message whereupon the upstream machine answers with its own ServiceDescription. This ServiceDescription message contains the lane ID and interface ID (optional) of the sending machine related to this TCP connection. It also contains a list of features which are implemented by the client. The features of the Hermes specification 1.0 have to be supported by any implementation and shall not be included explicitly.

If a downstream machine is already connected to the lane and the transportation interface, this connection will be retained. A Notification message shall be sent to the new connection before it is closed.

After exchanging the handshake messages, both machines may begin to send BoardAvailable/ MachineReady messages (see section 2.3.3).

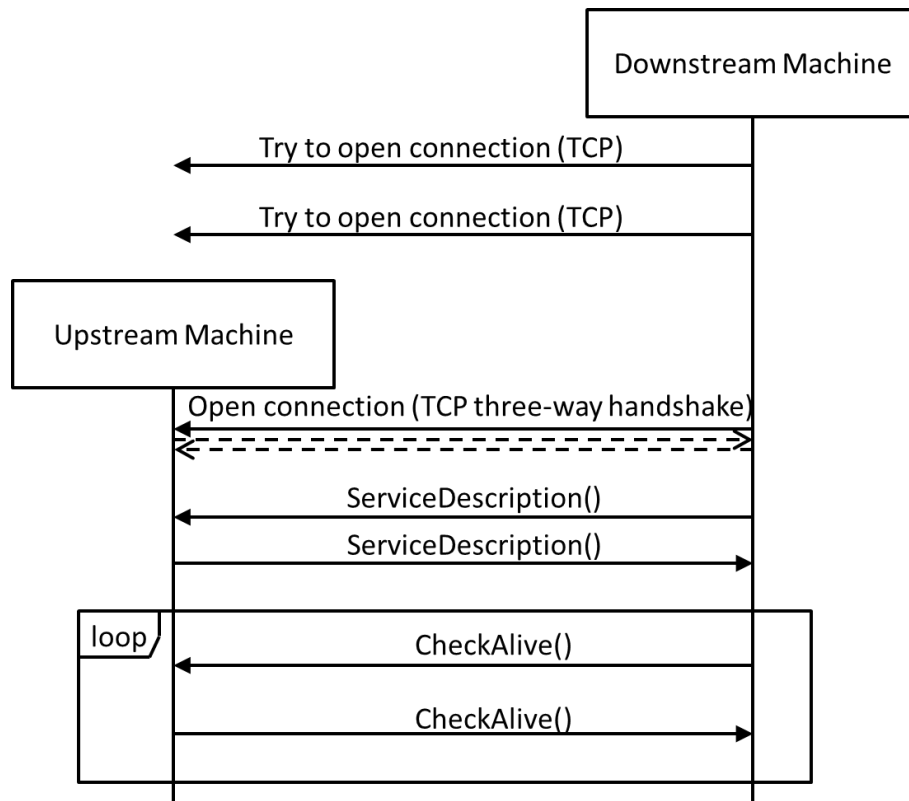


Fig. 4 Connection, handshake and connection loss detection on horizontal channel

The connections are kept open all the time. As TCP by itself does not detect connection losses ("Half-open connections" caused by e.g. process-/computer crash, unplugged network cables ...) both sides of a connection have to send cyclic CheckAlive messages. Those messages do not have to be answered by the remote side – the TCP stack will detect a connection loss when trying to send the packet. If the server detects a connection loss, it ends the connection and waits for a new connection by the client. If the client detects a connection loss, it ends the connection and re-starts with the cyclic connection attempts.

As not all TCP stacks recognize correctly the loss of connection when sending messages it is possible to extend the implementation of this functionality to an exchange of CheckAlive messages. Machines which have implemented this function do have the tag FeatureCheckAliveResponse in the ServiceDescription.

The exchange of CheckAlive messages then works like shown in Fig. 5.

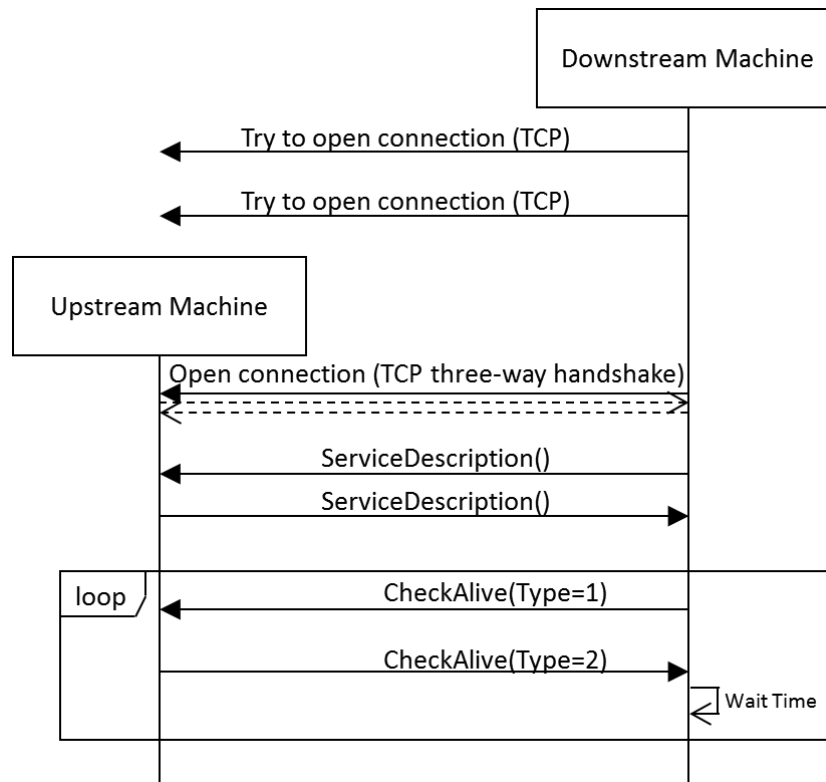


Fig. 5 Example for connection loss detection with FeatureCheckAliveResponse on horizontal channel

One of the machines (in the figure the downstream machine but it could be also the upstream machine) sends a (ping) CheckAlive message, that is a CheckAlive message with the attribute Type set to 1. The peer machine then responds immediately with a (pong) CheckAlive message, that is a CheckAlive message with the attribute Type set to 2 and the Id matching the Id of the (ping) CheckAlive message. A missing response (it is recommended to wait for 3 seconds.) indicates a connection loss.

2.3.3 Normal operation

When an upstream machine has a PCB available for handover, it sends a BoardAvailable message while a downstream machine ready to accept a PCB sends a MachineReady message. The naming of these messages is inspired by the electrical SMEMA interface. However, the messages do not represent the state of a machine's interface directly but are events for initiating a PCB handover.

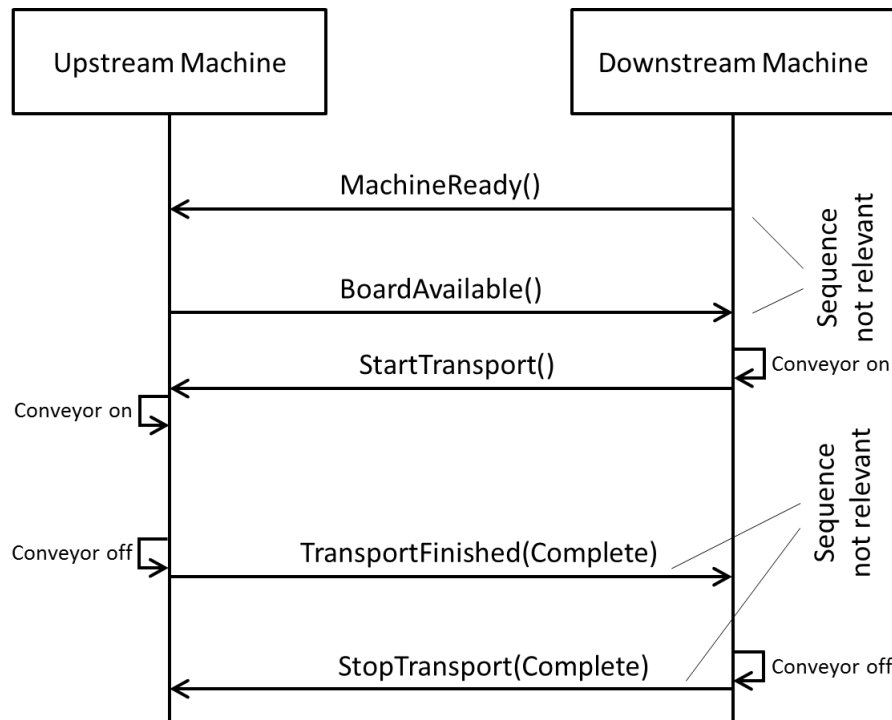


Fig. 6 Communication sequence for board transport

When both machines have indicated their readiness to handover the PCB, the downstream machine initiates the transfer by switching on its conveyor and sending the StartTransport message. Upon receiving this message, the upstream machine switches on its conveyor and the PCB moves into the downstream machine.

When the upstream machine is able to state that the PCB has fully left the machine, it sends the TransportFinished message. When the downstream machine has full control of the board, it sends the StopTransport message. The handover of a PCB is finished and is ready to start over.

If the upstream machine receives a StopTransport message and has not sent the TransportFinished message yet, it has to stop its conveyor and send the TransportFinished message.

The MachineReady message does not trigger an action on one of the machines directly. However it still is necessary to realize machines like e.g. shuttles which have to react to the availability of their downstream machines.

2.3.4 Transport error handling

To keep this protocol hardware independent, the handling of transport errors is described based on a very simple model of the board handover. The handover process is structured into the three phases

- 1 "NotStarted": The board is fully inside the upstream machine.
- 2 "Incomplete": The board is partly inside both machines.
- 3 "Complete": The board is fully inside the downstream machine.

Any state or event which prevents one or both machines from handing over a PCB is interpreted as an error. An error may be detected by any of the machines in any of the three handover phases. It is up to the

application how to detect the current handover phase, how to detect errors and how to solve them eventually (e.g. sensors, model based prediction, timeouts, user interaction ...).

The following sequence charts give an overview of the communication within this protocol depending on the machine which detects the error and the phase in which it is detected. The point in the sequence where the error is detected is marked by the following symbol: ●→^{Stop}

Scenario U1a

- Error detected by the upstream machine
- PCB fully inside the upstream machine
- Error detected before StartTransport has been received

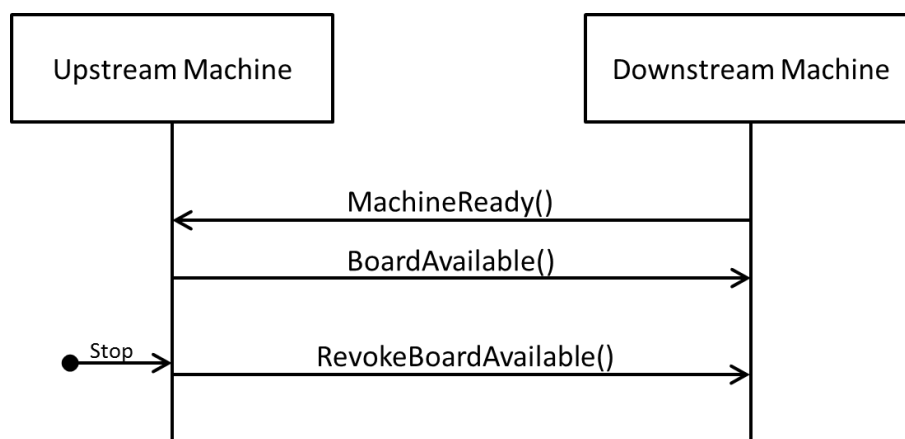


Fig. 7 Communication sequence in scenario U1a

Error detection: The error is detected before any transport started.

Reaction on upstream machine: The upstream machine sends a RevokeBoardAvailable message.

Reaction on downstream machine: None.

Resolution: After the error is solved, the regular transport sequence can start from the beginning.

Scenario U1b

- Error detected by the upstream machine
- PCB fully inside the upstream machine
- Error detected after StartTransport has been received

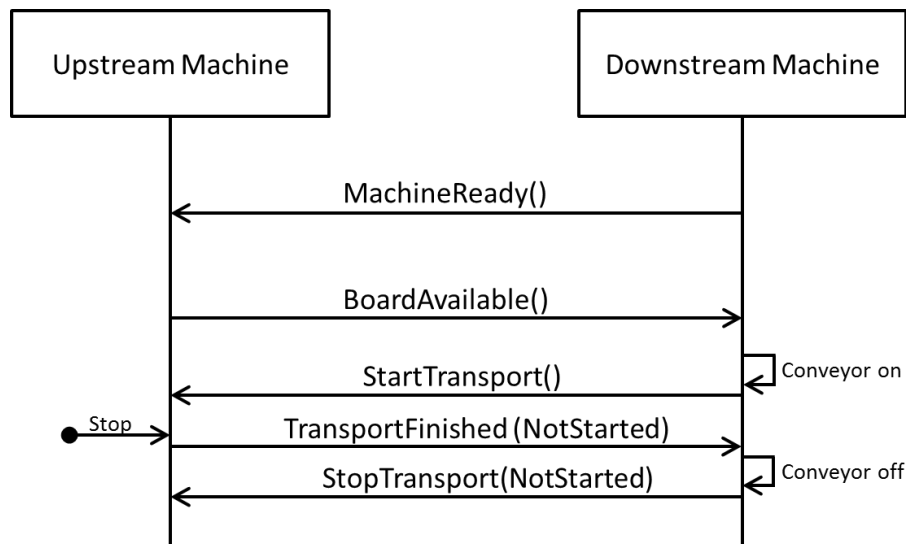


Fig. 8 Communication sequence in scenario U1b

Error detection: The error is detected after the downstream machine started its conveyor and has sent the `StartTransport` message.

Reaction on upstream machine: The upstream machine sends a `TransportFinished` message indicating that it has not started the transport.

Reaction on downstream machine: Upon the `TransportFinished` message, the downstream machine stops its conveyor and sends a `StopTransport` message indicating that no transport has started.

Resolution: After the error is solved, the regular transport sequence can start from the beginning.

Scenario U2

- Error detected by the upstream machine
- PCB partly inside both machines

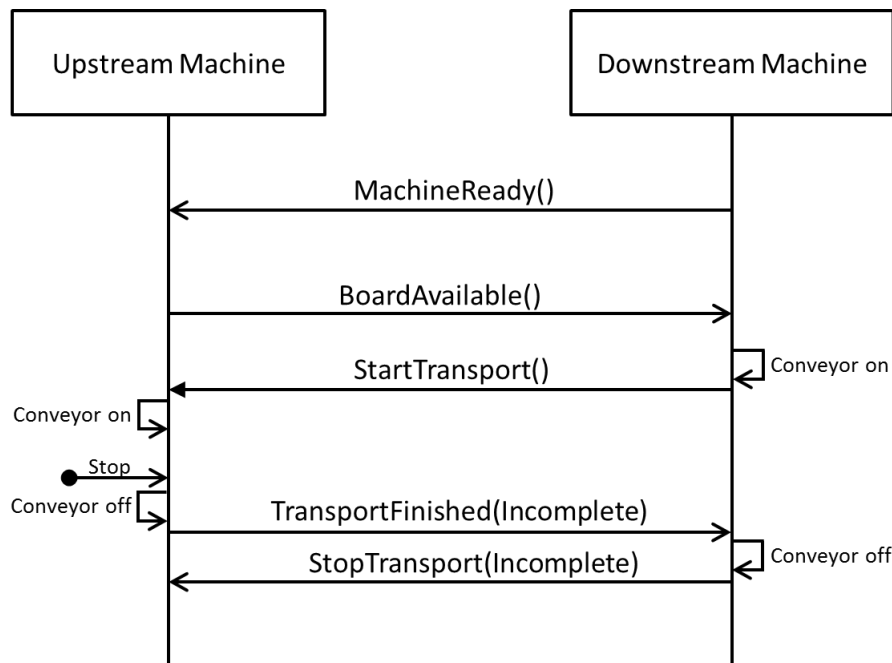


Fig. 9 Communication sequence in scenario U2

Error detection: The error is detected after both machines started their conveyors. The upstream machine assumes that the PCB may have partly entered the downstream machine.

Reaction on upstream machine: The upstream machine sends a TransportFinished message indicating that the PCB might be located between the machines.

Reaction on downstream machine: Upon the TransportFinished message, the downstream machine stops its conveyor and sends a StopTransport message indicating the state of the PCB handover. Note that in Fig. the StopTransport message is represented with parameter "Incomplete". However in this scenario, the downstream machine could send any of the allowed transport states.

Resolution: After the error is solved, the regular transport sequence can start from the beginning. The regular transport message sequence also applies to a PCB located between the two machines.

Scenario U3

- Error detected by the upstream machine
- PCB fully inside the downstream machine

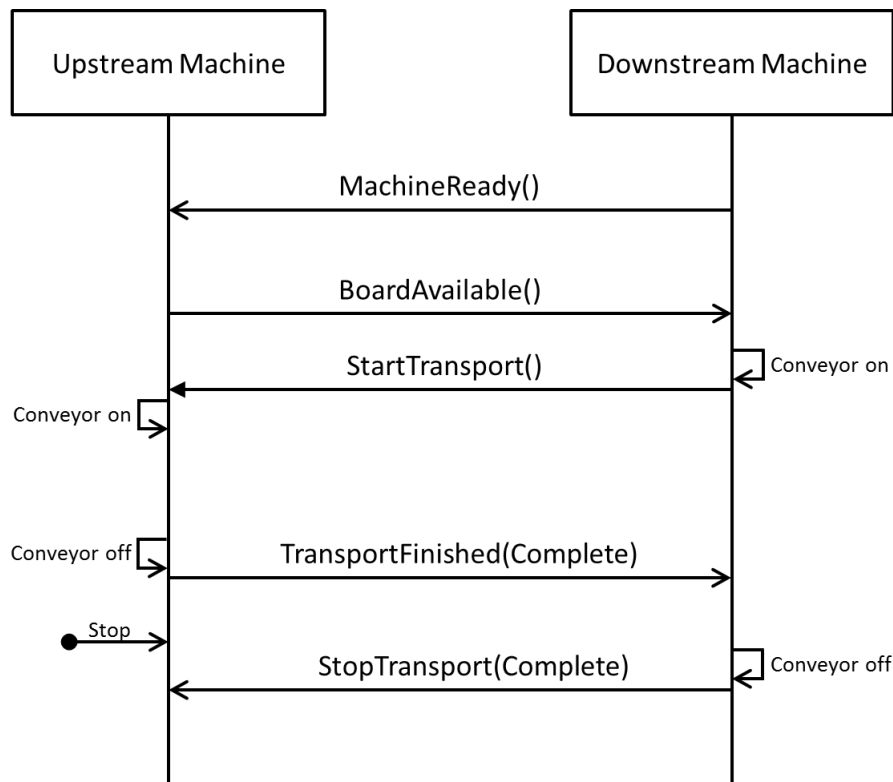


Fig. 10 Communication sequence in scenario U3

Error detection: The error is detected after the PCB is fully inside the downstream machine.

Reaction on upstream machine: None. Although the machine detected an error, it is irrelevant for the handover process.

Reaction on downstream machine: None. The downstream machine is not aware of any error.

Resolution: This scenario is irrelevant for the Hermes protocol. It is just listed for completeness.

Scenario D1

- Error detected by the downstream machine
- PCB fully inside the upstream machine
- Error detected before StartTransport has been sent

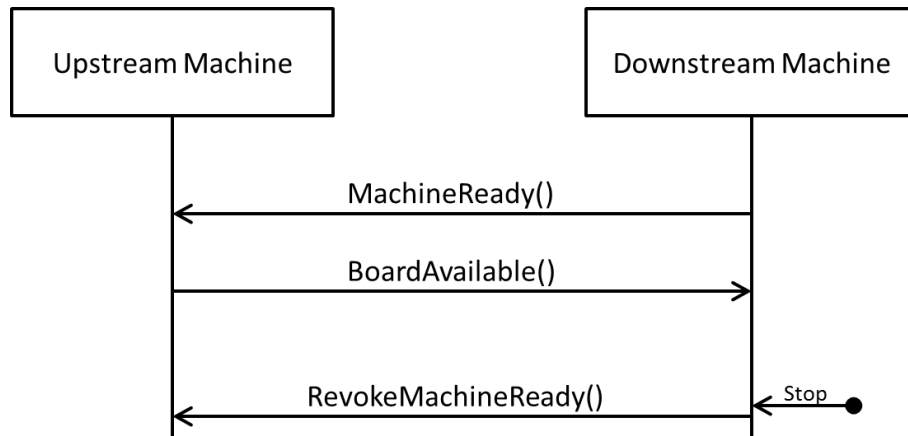


Fig. 11 Communication sequence in scenario D1

Error detection: The error is detected before any transport started.

Reaction on upstream machine: None.

Reaction on downstream machine: The downstream machine sends a RevokeMachineReady message.

Resolution: After the error is solved, the regular transport sequence can start from the beginning.

Scenario D2

- Error detected by the downstream machine
- PCB partly inside both machines

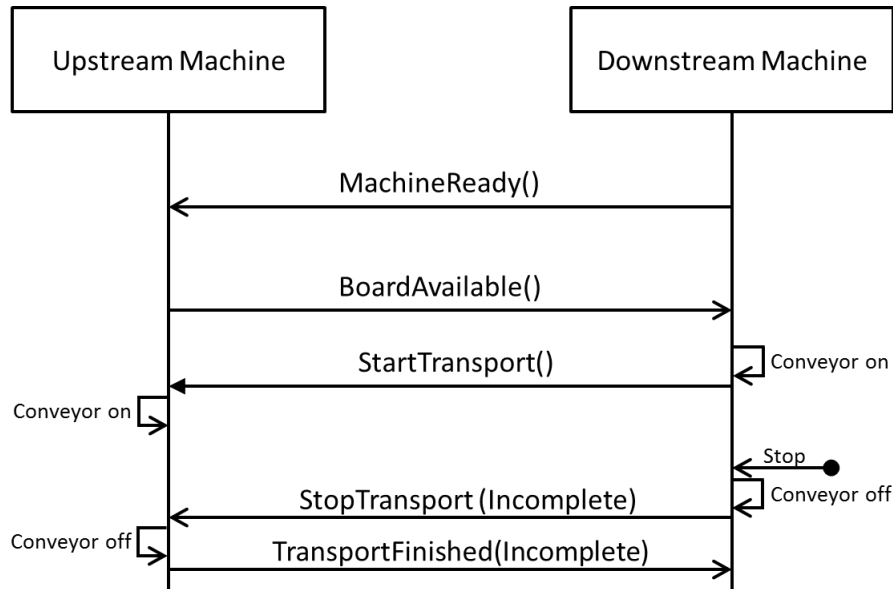


Fig. 12 Communication sequence in scenario D2

Error detection: The error is detected after both machines started their conveyors. The downstream machine assumes that the PCB may already have entered its conveyor.

Reaction on upstream machine: Upon the `StopTransport` message from the downstream machine, the upstream machine stops its conveyor and sends a `TransportFinished` message indicating the state of the PCB handover. Note that in Fig. 12 the `TransportFinished` message is represented with parameter "Incomplete". However in this scenario, the upstream machine could send any of the allowed transport states.

Reaction on downstream machine: The downstream machine stops its conveyor and notifies the upstream machine of the error by sending a `StopTransport` message indicating an incomplete PCB handover.

Resolution: After the error is solved, the regular transport sequence can start from the beginning. The regular transport message sequence also applies for a PCB located in between the two machines.

Scenario D3

- Error detected by the downstream machine
- PCB fully inside the downstream machine

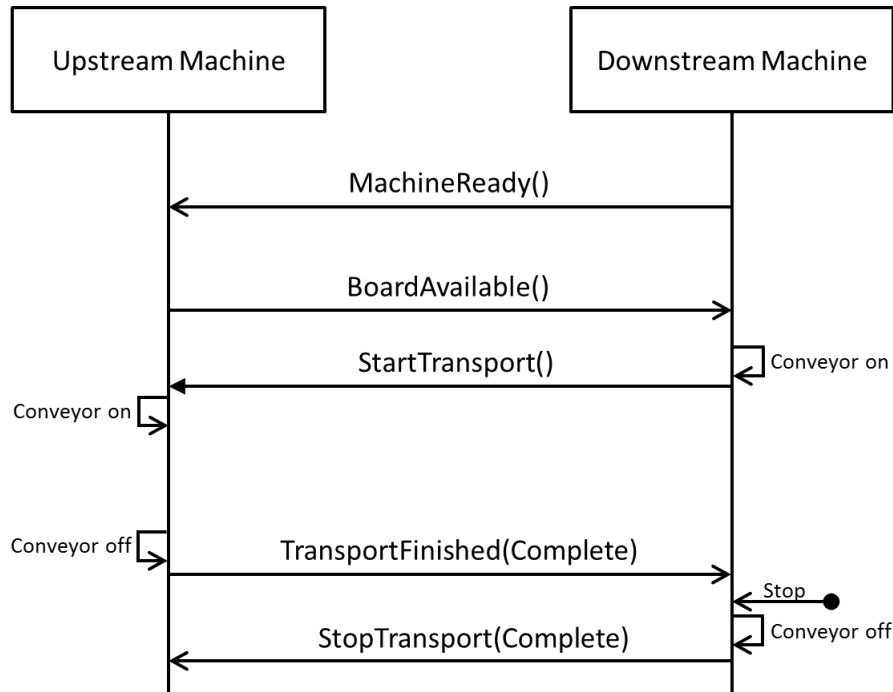


Fig. 13 Communication sequence in scenario D3

Error detection: The error is detected after the PCB is fully inside the downstream machine.

Reaction on upstream machine: None. The upstream machine is not aware of any error.

Reaction on downstream machine: None (at least in the scope of this protocol).

Resolution: This scenario is irrelevant for the Hermes protocol. As transport sequences are always initiated by the downstream machine sending `StartTransport`, trouble-shooting (possibly including running the conveyor of the downstream machine) can be executed independently from the upstream machine.

2.3.5 Handling of BoardForecast

Among others the BoardForecast may be used in following scenarios:

- Scenario 1: Anticipating a product change without a board (e.g. because upstream machine does not have stoppers / [band conveyor](#) that can be stopped).
- Scenario 2: Sending an estimated time to downstream machine until a board will be available (e.g. to allow downstream machine to choose between several upstream machines to get next available board).

Scenario 1:

Upstream machine is processing a changeover (new product type) and wants to ensure that the downstream machine is simultaneously also processing a changeover. Upstream machine also needs to check that this actually happens. It sends a BoardForecast with a (forecast-)ID, to which the downstream machine at some point must respond with a MachineReady with the same ID. Upon receiving this MachineReady, the upstream machine can assume that the product change was successful.

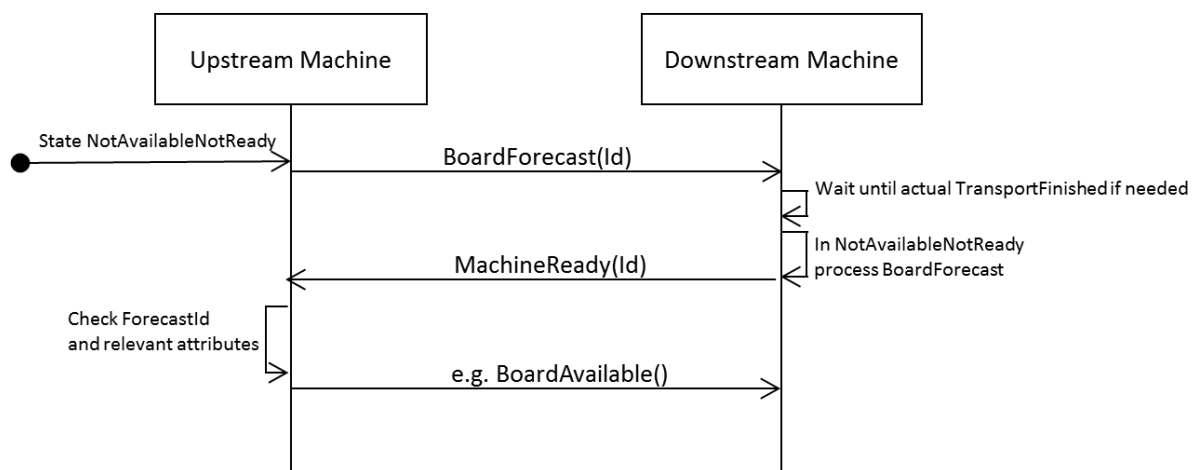


Fig. 14 Example of communication sequence for BoardForecast

Note: If starting the BoardForecast handling in the state MachineReady, the downstream machine must send a RevokeMachineReady message (see Fig. 15).

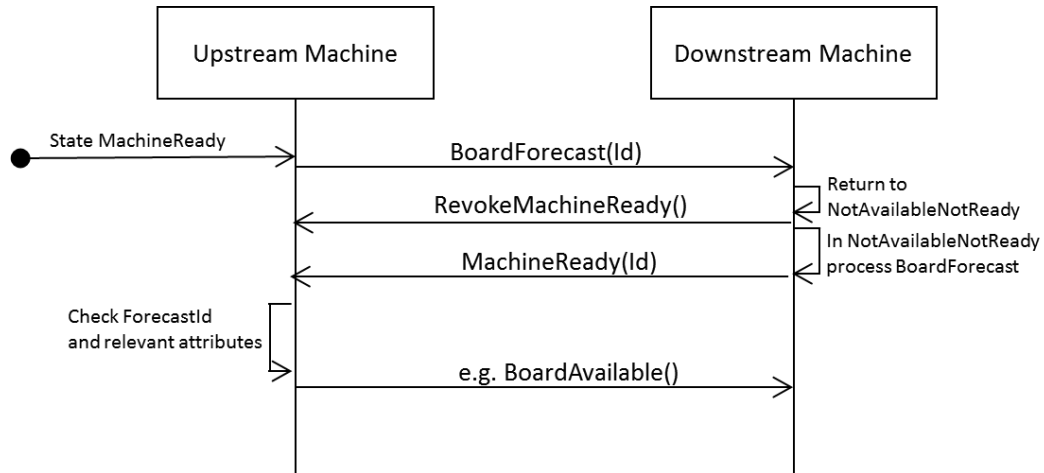


Fig. 15 Example of communication sequence for BoardForecast with RevokeMachineReady

If several BoardForecast messages (e.g. with different ProductTypeld) are sent in a short delay, the downstream machine may process only the last BoardForecast message:

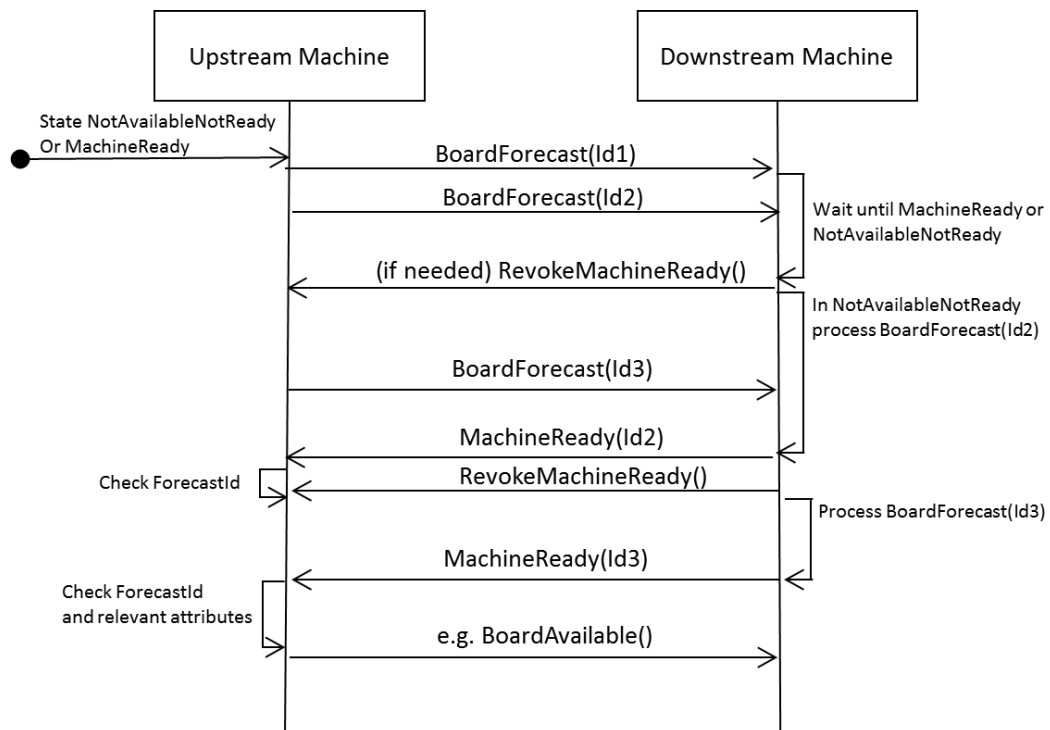


Fig. 16 Example of communication sequence with several BoardForecast

Scenario 1 (error handling):

If the downstream machine cannot accept the product exchange (e.g. unknown ProductId or width is physically impossible in machine) it will respond after a RevokeMachineReady with a notification of type "BoardForecastError". The upstream machine must then do some error [handling](#) (e.g. ask operator if machine should retry the BoardForecast or if the operator wants to remove the board).

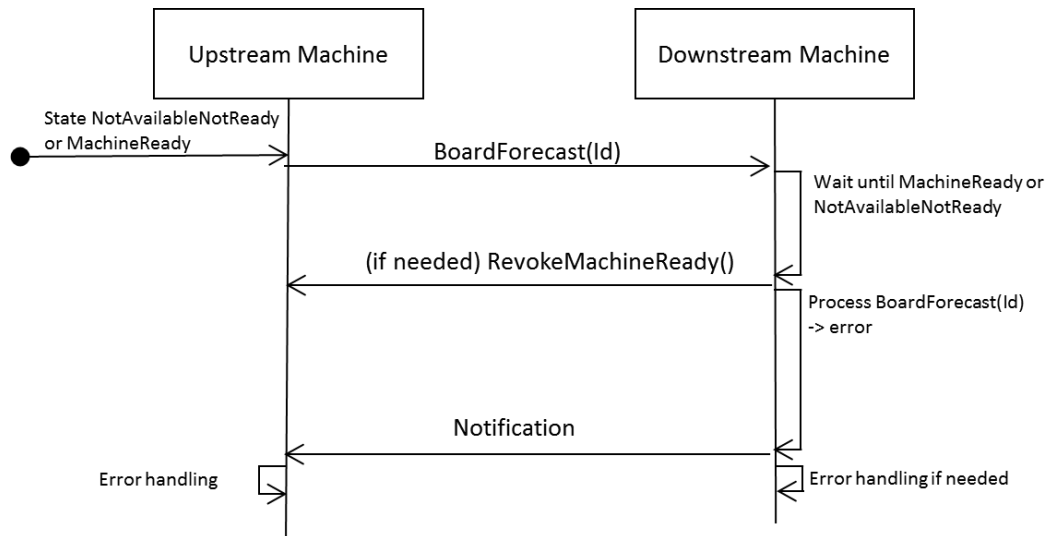


Fig. 17 Example of communication sequence in case with error handling

Scenario 2:

As BoardForecast in that case usually only gives some information to the downstream machine, several BoardForecast may be sent. However, error handling or checking are not needed on the side of the downstream machine. In that scenario ForecastId will not be sent.

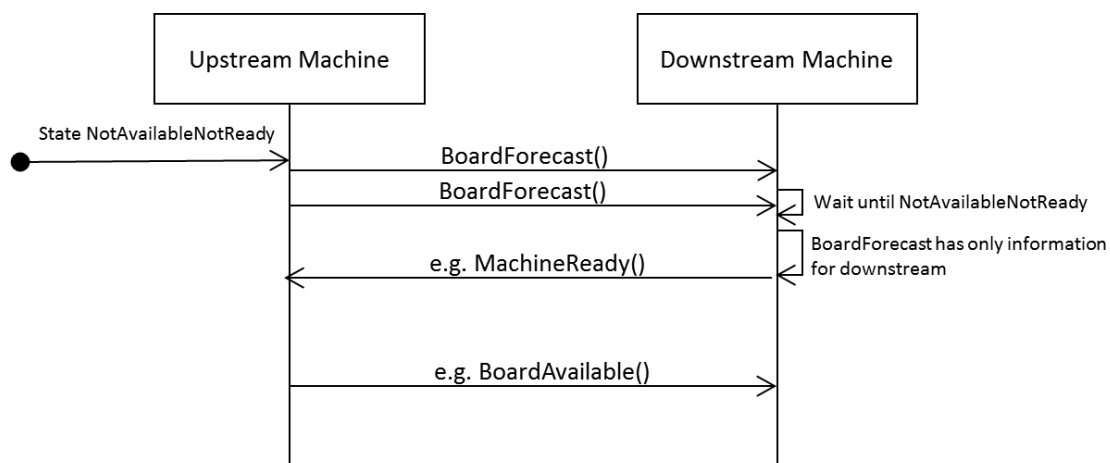


Fig. 18 Example of communication sequence BoardForecast without product change



Note: The function of BoardForecast is optional. If FeatureBoardForecast is specified in the ServiceDescription, it must be fully supported. Otherwise it can be ignored.

2.3.6 Protocol states and protocol error handling

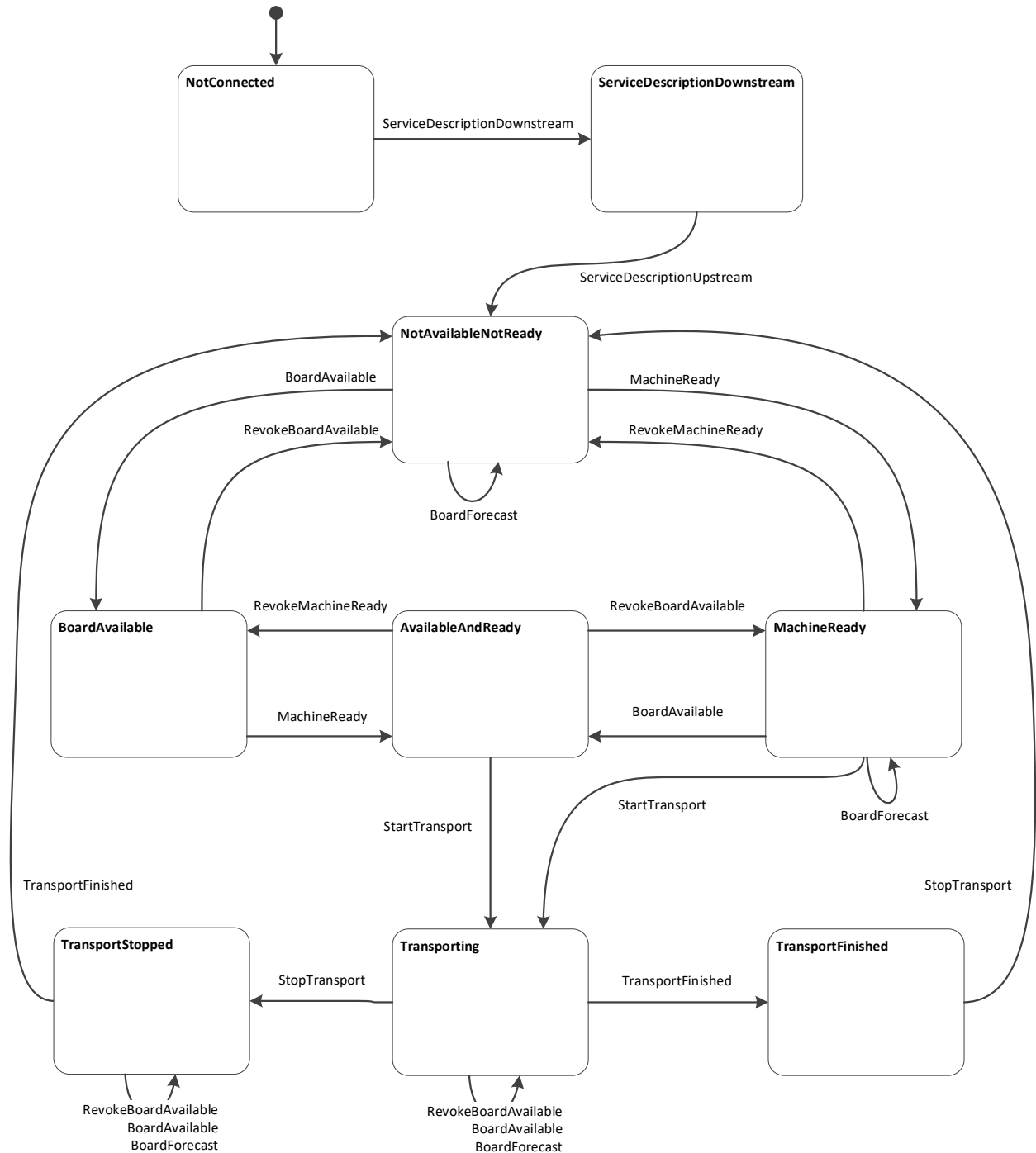


Fig. 19 Hermes interface states on horizontal channel

Fig. 19 lists all states and transitions of a Hermes interface corresponding to the machine-to-machine (M2M) communication. The state is the comprehensive state of the interface rather than the state of one of the involved machines.



The messages may only be sent if they trigger the corresponding transition shown in the state chart. Any message **defined in this standard**, except "Notification", "CheckAlive", "QueryBoardInfo" and "SendBoardInfo", which is received not triggering a transition is interpreted as a protocol error (e.g. a MachineReady message when the interface is in the state Transporting). In case of a protocol error, any running transport shall be stopped and the connection is terminated. The interface may start over with a new connection. **Any unknown message, which is received, shall be ignored and discarded to keep upward compatibility.**

Note that due to race conditions, a RevokeBoardAvailable message may overlap with a StartTransport message or even a StopTransport message, so this shall not be treated as a protocol error (transition from MachineReady to Transporting and self-transitions on Transporting and TransportStopped).

2.4 Remote configuration

2.4.1 Topology

Although a machine may offer the possibility to configure the Hermes TCP port(s) and the IP address(es) of its upstream machine(s) locally (e.g. via a graphical user interface of the machine controller), every machine implementing this protocol shall offer a possibility to configure these properties remote via TCP. Therefore, the machine shall offer a TCP server on port 1248 on at least one network adapter where it accepts configuration messages (see sections 3.19 to 3.21 for detailed information).

The configuration system opens a connection to each required machine. The connection shall only be kept open as long as needed and closed by the configuration system.

2.4.2 Remote configuration

A SetConfiguration message shall contain the full configuration for all Hermes interfaces of a machine. Any existing configuration is overwritten when a SetConfiguration message is received. Whenever a configuration is not applicable (e.g. bad IP address format), the SetConfiguration message is answered with a Notification message (see section 3.5). Every time the configuration is changed, **affected** open Hermes connections will be reset at the next appropriate moment.

It is possible to read the current configuration through the GetConfiguration message answered by a CurrentConfiguration message. The configuration shall be persisted until it is changed.

2.5 Communication with supervisory system (vertical channel)

2.5.1 Topology

Any machine in a line shall offer one TCP server on the configured supervisory system port on at least one network adapter where it accepts connections from supervisory systems. The used supervisory system port can be retrieved via GetConfiguration. The connection to the supervisory system is e.g. used to allow the configuration of the Hermes connections to the upstream and downstream machine(s) remotely without relying on the capabilities of the machine user interface.

The supervisory system opens a connection to each required machine. The connection shall only be kept open as long as needed and closed by the supervisory system.

Note: It is possible to use the same port for the communication with a supervisory system as for the remote configuration.

2.5.2 Connecting, handshake and detection of connection loss

Upon demand the supervisory system starts cyclic connection attempts to the required machine. When a connection is established, the supervisory system starts sending a SupervisoryServiceDescription message whereupon the machine answers with its own SupervisoryServiceDescription. This SupervisoryServiceDescription message contains a list of supervisory features which are implemented by the client.

If a new supervisory system tries to connect and no further connections are supported by the machine, the already established connections will be retained. A Notification message shall be sent to the new connection before it is closed.

After exchanging the handshake messages, both communication partners may begin to exchange the messages belonging to supervisory features supported by both communication partners.

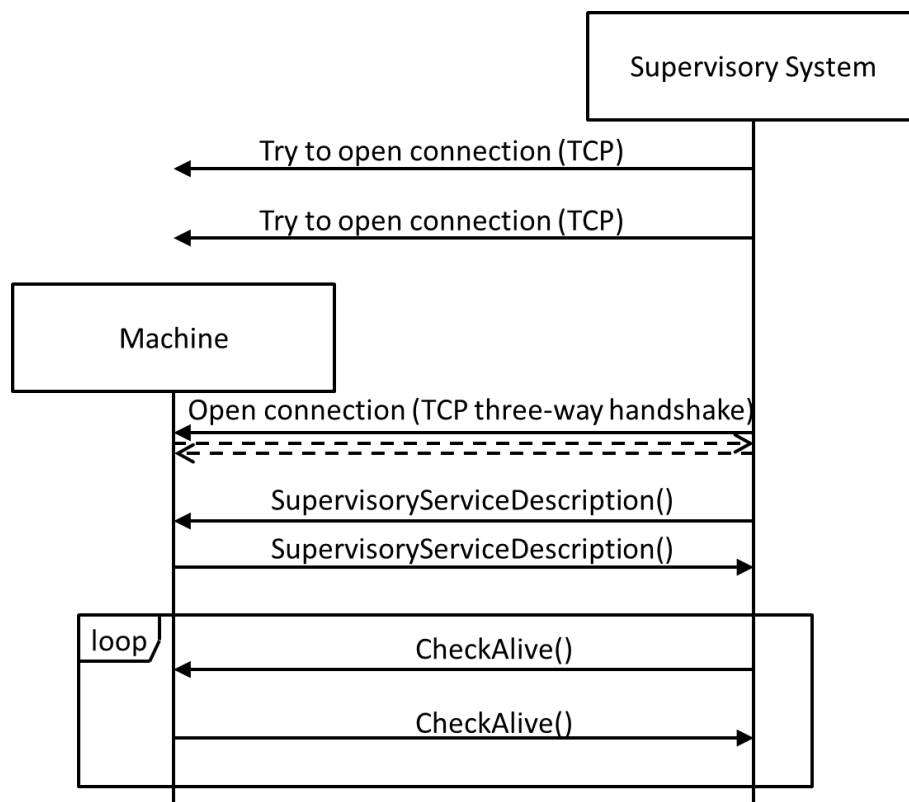


Fig. 20 Connection, handshake and connection loss detection on vertical channel

The connections are kept open as long as needed. As TCP by itself does not detect connection losses ("half-open connections" caused by e.g. process- / computer crash, unplugged network cables ...) both sides of a connection have to send cyclic CheckAlive messages. Those messages do not have to be answered by the

remote side – the TCP stack will detect a connection loss when trying to send the packet. If the server detects a connection loss, it ends the connection and waits for a new connection by the client. If the client detects a connection loss, it ends the connection and re-starts with cyclic connection attempts.

As not all TCP stacks recognize correctly the loss of connection when sending messages it is possible to extend the implementation of this functionality to an exchange of CheckAlive messages. Machines which have implemented this function do have the tag `FeatureCheckAliveResponse` in the `SupervisoryServiceDescription`.

The exchange of CheckAlive messages then works like shown in Fig. 21.

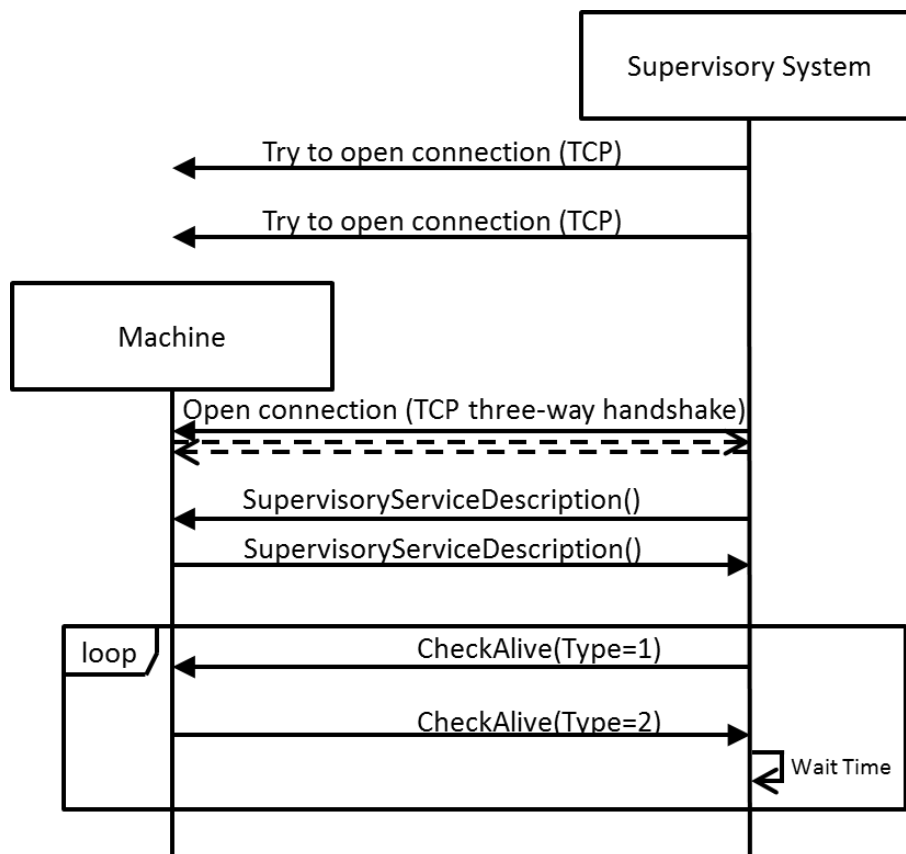


Fig. 21 Example for connection loss detection with `FeatureCheckAliveResponse` on vertical channel

One of the communication partners (in the figure the supervisory system but it could be also the machine) sends a (ping) CheckAlive message, that is a CheckAlive message with the attribute Type set to 1. The peer communication partner then responds immediately with a (pong) CheckAlive message, that is a CheckAlive message with the attribute Type set to 2 and the Id matching the Id of the (ping) CheckAlive message. A missing response (It is recommended to wait for 3 seconds.) indicates a connection loss.

2.5.3 Protocol states and protocol error handling

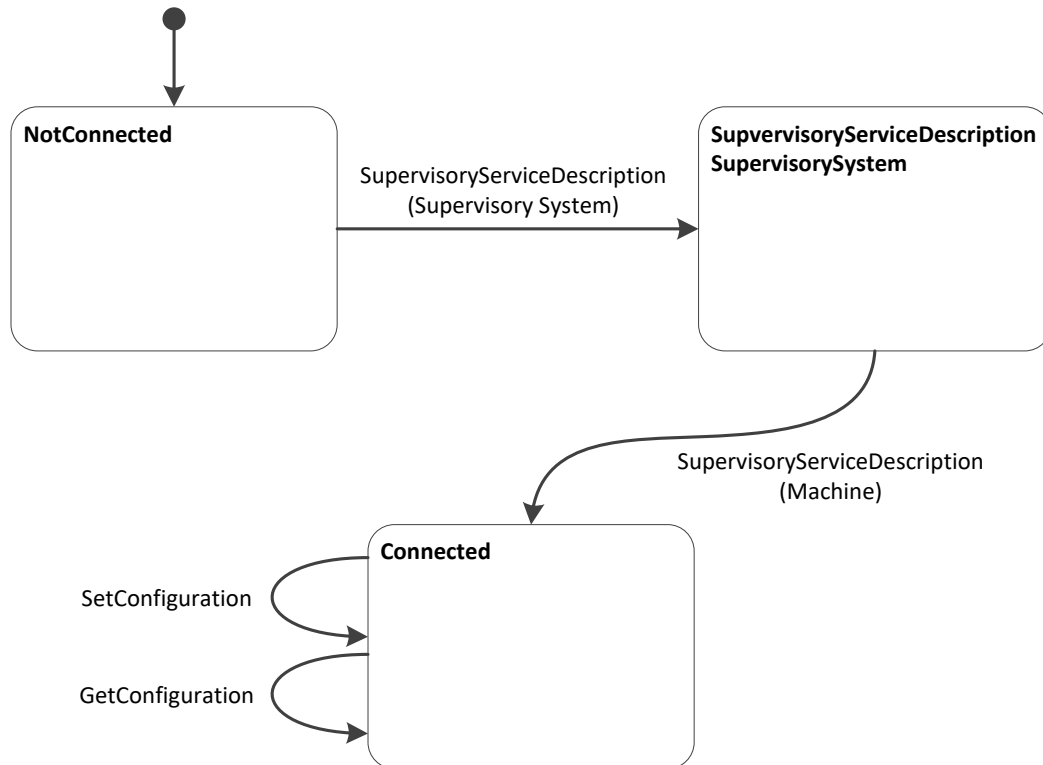


Fig. 22 Hermes interface states on vertical channel

Fig. 22 lists all states and transitions of a Hermes interface corresponding to the communication with supervisory systems. The state is the comprehensive state of the interface rather than the state of one of the involved communication partners.

The messages may only be sent if they trigger the corresponding transition shown in the state chart. Any message defined in this standard, except "Notification" and "CheckAlive", which is received not triggering a transition is interpreted as a protocol error. In case of a protocol error the connection is terminated. The interface may start over with a new connection. Any unknown message, which is received, shall be ignored and discarded to keep upward compatibility.

3 Message definition

...

3.5 Notification

The Notification message is sent by both machines before a connection is terminated, e.g. after protocol errors or before shutdown. It could also be used for general notification purposes.



Notification	Type	Range	Optional	Description
◆ NotificationCode	int	1 .. n	no	A notification code of the list below. Notification codes above 1000 are not defined by this protocol and may be used by the application
◆ Severity	int	1 .. 4	no	A severity level from of the list below
◆ Description	string	any string (minimum supported length: 254 bytes)	no	An English textual description of the notification.

The following NotificationCodes are defined:

- 1 Protocol error (invalid transition in the corresponding state machine, see section 2.3.4)
- 2 Connection refused because of an established connection
- 3 Connection reset because of changed configuration
- 4 Configuration error
- 5 Machine shutdown
- 6 BoardForecast error

Possible values for Severity:

- 1 Fatal error
- 2 Error
- 3 Warning
- 4 Info



3.19 SetConfiguration





The SetConfiguration message is sent by an engineering station to configure the Hermes interfaces of a machine. If the sent configuration is not accepted, the machine is expected to send a Notification message (see section 3.5).

Note: The function of SetConfiguration is optional on the vertical channel. If FeatureConfiguration is specified in the SupervisoryServiceDescription, it must be fully supported. Otherwise it can be ignored.

SetConfiguration	Type	Range/ Multiplicity	Optional	Description
◆ MachineId	String	any string (minimum supported length: 80 bytes)	No	ID/name of this machine for identifying it in a Hermes enabled production line.
◆ SupervisorySystemPort	int	0 .. 65535	yes	Port number on



				which connections from supervisory systems shall be established
 UpstreamConfigurations	UpstreamConfiguration []	0 .. n	No	Configuration for upstream lanes
 DownstreamConfigurations	DownstreamConfiguration []	0 .. n	No	Configuration for downstream lanes

UpstreamConfiguration	Type	Range/ Multiplicity	Optional	Description
 UpstreamLaneId	int	1 .. n	no	The lane on the upstream side Lanes are enumerated looking downstream from right to left beginning with 1
 UpstreamInterfaceId	string	any string (minimum supported length: 80 bytes)	yes	The ID of the transportation interface on the upstream side
 HostAddress	string	valid IP address or hostname (minimum supported length: 254 bytes)	no	The IP address or hostname of the upstream machine for this lane and transportation interface
 Port	int	0 .. 65535	no	Port number on which connections shall be established



DownstreamConfiguration	Type	Range/ Multiplicity	Optional	Description
◆ DownstreamLaneId	int	1 .. n	no	The lane on the downstream side Lanes are enumerated looking downstream from right to left beginning with 1
◆ DownstreamInterfaceId	string	any string (minimum supported length: 80 bytes)	yes	The ID of the transportation interface on the downstream side
◆ ClientAddress	string	valid IP address or hostname (minimum supported length: 254 bytes)	yes	The IP address or hostname of the downstream machine for this lane and transportation interface. If not specified, then connections from any IP address are accepted.
◆ Port	int	0 .. 65535	no	Port number on which the server shall accept connections for this lane

All connections where the machine is acting as board provider are stored in DownstreamConfigurations. All connections where the machine is acting as board receiver are stored in UpstreamConfigurations. These are independent of the board transport direction of the SMT line.

It is up to the user to keep MachineIds unique.

3.20 GetConfiguration

The GetConfiguration message is sent by an engineering station to read out the current configuration of the Hermes interfaces of a machine. The machine is expected to answer with a CurrentConfiguration message.

Note: The function of GetConfiguration is optional on the vertical channel. If FeatureConfiguration is specified in the SupervisoryServiceDescription, it must be fully supported. Otherwise it can be ignored.




GetConfiguration	Type	Range/ Multiplicity	Optional	Description
------------------	------	---------------------	----------	-------------

3.21 CurrentConfiguration

The CurrentConfiguration message is sent by a machine in response to the GetConfiguration message.

CurrentConfiguration	Type	Range/ Multiplicity	Optional	Description
◆ MachineId	string	any string (minimum supported length: 80 bytes)	yes	ID/name of this machine for identifying it in a Hermes enabled






		80 bytes)		production line.
 SupervisorySystemPort	int	0 .. 65535	yes	Port number on which connections from supervisory systems shall be established
 UpstreamConfigurations	UpstreamConfiguration []	0 .. n	No	Configuration of upstream lanes
 DownstreamConfigurations	DownstreamConfiguration []	0 .. n	No	Configuration of downstream lanes



For the definition of UpstreamConfiguration and DownstreamConfiguration see section 3.19.

If no MachineId has been configured yet, the CurrentConfiguration message does not contain the attribute MachineId.

3.22 SupervisoryServiceDescription

The SupervisoryServiceDescription message is sent by both machine and supervisory system after a connection is established. The supervisory system sends its SupervisoryServiceDescription first whereupon the machine answers by sending its own SupervisoryServiceDescription.

SupervisoryServiceDescription	Type	Range	Optional	Description
 SystemId	String	any string (minimum supported length: 80 bytes)	no	ID / name of the sending machine or supervisory system for identifying it in a Hermes enabled production line.
 Version	String	xxx.yyy (7 bytes)	no	The implemented interface version of the machine or supervisory system
 SupportedFeatures	SupervisoryFeature []		no	List of supported supervisory features (empty for version 1.0)

SupervisoryFeature	Type	Range	Optional	Description
 FeatureConfiguration	FeatureConfiguration		yes	Indication of configuration functions implementation
 FeatureCheckAliveResponse	FeatureCheckAliveResponse		yes	Indication of CheckAliveResponse function implementation

xxx.yyy must match the regular expression

`[1-9][0-9]{0,2}\.[0-9]{1,3}`

