UNISIG

ERTMS/ETCS	

STM FFFIS Safe Time Layer

REF : SUBSET-056

ISSUE : 3.0.0

DATE : 2012-02-29

Company	Technical Approval	Management approval
ALSTOM		
ANSALDO STS		
BOMBARDIER		
INVENSYS RAIL		
SIEMENS		
THALES		

1. MODIFICATION HISTORY

Issue Number Date	Section Number	Modification / Description	Author
0.0.1 25-11-99		First draft	J. Näsström
0.0.2 01-12-99		Update after STM meeting, see ATD_MIN_991130_WGSTM	J. Näsström
0.0.3 03-01-00		New structure of the document. • Added version control • Separate messages for time sync. And time reference. • Minor improvements in the whole document. • Editorial updates.	N. Lindström
0.0.4 25-01-00		Update after STM meeting 20-01-00 • Minor improvements in the whole document. • Editorial updates. • Removed application specific items. Application specific items shall be included in the application document. • The Sync and Reference Time messages are combined into one message. • Major improvements in Section.	N. Lindström
0.0.5 07-02-00		Update after comments after comments from the STM group • General improvements of section 7. • Minor improvements in the whole document. • Version control moved to Safe Link Layer.	N. Lindström
0.0.6 11-02-00		Added a new message 'Boadcast startup'. • Improvements on clock synchronisation using redundant buses. • Added version control of broadcast messages.	N. Lindström
1.0.0 21-02-00		Update after comments from Unisig STM meeting 17-02-00 in Stuttgart. • Moved section 'Logical Addressing' to the STM FFFIS main document. • Added version control of the Sate time Layer. • Several clarification in Section. • Added appendix about provided services.	N. Lindström
1.0.1 28-03-00		Update after comments from Unisig STM meeting 23-03-00 in Paris and CENELEC meeting 09-03-00. • Harmonised several definitions in section 3.2 with definitions in Safe Link Layer. • Removed section 4.1 (It is moved to FFFIS STM) • Added definition of disconnect_reason for Disconnect Telegram , see section 4.3 • Adaptations to the new redundancy concept. • Removed Bus Time. • Tolerance of long term drift changed from 1% to 0.1%. (purposed by CENELEC) • Improved criterion for long time drift supervision (purposed by CENELEC). • Adaptations in order to let the application disconnect and reconnect a logical Connection.	N. Lindström

Issue Number Date	**Section Number**	**Modification / Description**	**Author**
2.0.0 30-03-00	Ref.	3.1.1.2, 8.1.1.3, 8.1.1.5 Final Issue to ECSAG	D. Degavre (Ed)
2.0.1 23-01-02		Update after comments from Unisig STM meeting 04-12-01 in Brussels. • Some definitions moved to section 3.2. • Major changes in order to add general clarifications. • Req 7.2.1.4, 7.4.1.1, 7.4.1.2 and 7.4.1.2.1, Added.	J. Näsström
2.0.2 06-02-02		Update after comments from Unisig STM meeting 22-01-02 in Sttutgart.	B. Muñoz
2.0.3 14-03-02		Update after comments from Unisig STM meeting 14-03-02 in Berlin.	B. Muñoz
2.0.4 20-03-02		Update after pending comments from Unisig STM meeting 14-03-02 in Berlin.	B. Muñoz
2.0.5 12-04-02		Update after comments from Unisig STM meeting 12-04-02 in Stuttgart	B. Muñoz
2.0.6 2002-APR-18		Update after pending comments from Unisig STM meeting 2002-APR-14 in Stuttgart.	B. Muñoz
2.0.7 2002-MAY-14		Update after comments from Unisig STM meeting 2002-MAY-14 in Brussels.	B. Muñoz
2.0.8 2002-MAY-21		Update after pending comments from Unisig STM meeting 2002-MAY-14 in Brussels.	B. Muñoz
2.0.9 2002-JUN-11		Update after comments from Unisig STM meeting 2002-06-11 in Braunschweig.	P. Luehrs
2.0.10 2002-JUN-18		Update after pending comments from Unisig STM meeting 2002-JUN-11 in Braunschweig.	B. Muñoz
2.0.11 2002-JUN-25		Update after pending comments from Unisig STM meeting 2002-JUN-25 in Madrid.	R.Ramos
2.1.0 2002-JUN-27		Editorial Changes	R. Ramos
2.2.0 2003-06-19	5.3	Req. about non-final disconnection for SL 0 connections (former 5.3.1.4) deleted	P. Lührs (Siemens)
	2	Table of Contents updated	
2.3.0 20-12-17	6.1.1.3	WG-1: Command value “8Dh” added for Multicast.	P. Luehrs (Siemens)
	7.4.3.7, 7.4.3.7.1	WG-2: Formula for the check of the long term drift corrected and explanation for the check reworded.	
	8.2	Compatibility Number changed to 3.1.Z, as the compatibility is not affected.	
	7.4.2.2 7.4.2.2.1 7.4.2.2.2	WG-6: Requirements added for the first synchronisation	R. Ramos (Invensys Rail)
		Update to new UNISIG document external template 2009	A.Geck
	3.4.10	WG-15: Add missing definitions for receiver transfer times	(Bombardier)
	3.4.10	WG-15: Editorial corrections during meeting	A.Geck (Bombardier)
	front page	WG-17:Removed names of associated members and change ‘ANSALDO SIGNAL’ to ‘ANSALDO STS’	A.Geck (Bombardier)
	7.8.1.14.4	WG-5: Error handling for multicast messages	
	6.6.1.2.1	WG-7: No MC message before clock synchronisation	
	6.6.1.2.2	WG-8: No MC message before STL startup	

Issue Number Date	Section Number	Modification / Description	Author
	7.2.1.8deleted	WG-9+14: Reference Clock in Clock Master	A.Geck (Bombardier)
	3.4.8.1.2 7.2.1.19 7.2.1.20 7.4.5.1 7.4.5.2 7.4.5.2.1 7.4.5.2.2	WG-9: Reference Clock in Clock Master	
	7.4.3.7.2.2	WG-10: Accuracy of Reference Clock	
	9.6.1.2 3.4.10.2	WG-13: Sender LCI	
	1 front page 8	Corrections in version history and spelling corrections Version and date changed	A.Geck (Bombardier)
	7.7.1.2 7.7.1.2.1 7.7.1.2.2	WG-11:Handling of 'connection established by SLL' indication when the clock synchronisation has been lost	
2.9.1 2012-01-21		Update according to ERA remarks	A.Geck (Bombardier)
	whole document	Comment #1, #5 and #9 (using shall in requirements)	
	5.3, 7.6, 7.7, 10.1.1.2.1	Comment #2(disconnect reason)	
	3.4.10.1.2 3.4.10.2.2 3.4.10.5 7.2.1.19	Comment #3 (token rotation time explanation)	
	4.1.1.3 4.1.1.4 Deleted	Comment #4 (reference clock)	
	6.1.1.2 6.1.1.5	Comment #6 (command numbers)	
	6.6.1.3, 6.6.1.4, 7.3.1.4, 7.3.1.5,7.4.1.6.1, 7.4.3.5, 7.4.3.6, 9	Comment #7 (definition of constants)	
	7.1.1.2.4 deleted	Comment #8 (implementation hints)	

Issue Number Date	Section Number	Modification / Description	Author
	7.4.3.7.2 deleted	Comment #10 (redundant clause)	
	3.3.1.4, 12 deleted	Comment #12 (Metaphor)	
	1 note deleted, 3.3.1.3 5.2.1.3 5.3.1.3 6.3.1.9 6.6.1.5 6.1.1.6 8 deleted	CR 1043 (ERA comment #11)	
		Consolidation 20120116 (acc. to WG STM meeting)	
	4.2.1.2	Time stamps used on application level	
	Whole document	Added titles to all figures and updated references to them	
	Whole doc., mainly 7.2, 7.4	Cleanup usage of terms for local time and reference time	
2.9.2 2012-02-17	Front page	Update according to ERA/SG review remarks for v2.9.1: Removed ‘Baseline 3’ in top line	
		Update according to SG review remarks for v2.9.1:	
	1	Remark 2 (inconsistencies in version history): V2.3.0: incorrect link updated V2.9.1:consolidation date changed	
	3.1	Remark 3: List of references changed to table	
	All	Updated links to references	
3.0.0 2012-02-29	No change	Baseline 3 release version	Thomas Mandry (Alstom)

2. Table of Contents

1. MODIFICATION HISTORY 2
2. TABLE OF CONTENTS 6
3. GENERAL 8
   3.1 References 8
   3.2 Abbreviations 8
   3.3 Scope 8
   3.4 Definitions 8
      3.4.1 General 8
      3.4.2 Logical connection Master 9
      3.4.3 Logical connection Slave 9
      3.4.4 Multicast 9
      3.4.5 Reference Clock 9
      3.4.6 Reference Time 10
      3.4.7 Local Clock 10
      3.4.8 Local Reference Time 10
      3.4.9 STL_TIME_STAMP 11
      3.4.10 Transfer Times 11
4. SUMMARY DESCRIPTION 13
   4.2 Time stamps 14
5. SAFE TIME LAYER CONTROL 16
   5.1 Start-up of Safe Time Layer 16
   5.2 Configuration data of Safe Time Layer 18
   5.3 Disconnect Reason and Type 18
   5.4 STL_Time_Stamp 19
6. MESSAGES 20
   6.2 Application Data 21
   6.3 Sync and Reference Time 21
   6.4 Ready to Run 22
   6.5 Run 22
   6.6 Safe Time Layer Startup for multicast 23
7. SPECIFICATION OF FUNCTIONS 25
   7.1 GENERAL 25
   7.2 Principle of Clock Synchronisation 26
   7.3 Reference Clock 28

7.4 Local Clock 29
7.4.1 General 29
7.4.2 Synchronise 31
7.4.3 Resynchronise 31
7.4.4 Calculation of Adjustment Factor 33
7.4.5 Calculation of Local_Reference_Time 33
7.5 Principle of Logical Connection 34
7.5.1 General for Point-to-point connection 34
7.6 Logical Connection Master 34
7.7 Logical Connection Slave 37
7.8 Time Validation of Received Messages 39
8. INTENTIONALLY DELETED 41
9. LIST OF CONSTANTS 42
9.2 LocalClockMaxReSyncInterval 42
9.3 SafeTimeLayerStartupInterval 42
9.4 SafeTimeLayerReStartInterval 42
9.5 StartupSynchronisationTimeLimit 42
9.6 MaxClockInaccuracyAfterAdjustFactor 42
9.7 TimeForLongTermDriftCheck 42
9.8 MinNumberofSyncAndRefMsgReceived 43
9.9 SyncAndRefTimeSyncInterval 43
9.10 SyncAndRefTimeRunInterval 43
9.11 SyncAndRefTimeStartupTimeLimit 43
9.12 ConnectionSetupTimeLimit 43
10. APPENDIX: SERVICES PROVIDED BY THE SAFE TIME LAYER 44
11. APPENDIX: DEFINITION OF NOTATION 45
11.1 State Machine 45
11.2 Sequence Chart 45
12. INTENTIONALLY DELETED 47

3. GENERAL

3.1 References

Ref. N°	Document Reference	Title
/1/	CENELEC EN 50170-2 (1996)	PROFIBUS
/2/	SUBSET-057	STM FFFIS Safe Link Layer
/3/	SUBSET-059	Performance requirements for STMs

3.2 Abbreviations

3.2.1.1	SLL	Safe Link Layer /2/
3.2.1.2	STL	Safe Time Layer
3.2.1.3	Profibus	Process Field Bus
3.2.1.4	FDL	Field Data Link /1/
3.2.1.5	SAP	Service Access Point /1/

3.3 Scope

3.3.1.1 This document specifies the Safe Time Layer.

3.3.1.2 Together with Safe Link Layer /2/ it will provide safe communication with protection against the top hazards of communication (sequence faults, undetected data corruption, data ageing faults and authenticity faults).

3.3.1.2.1 Data ageing in the sending computer before reaching the Time Layer is not part of this document.

3.3.1.3 The Safe Time Layer also provides solutions for clock synchronisation and failure monitoring.

3.3.1.4 Intentionally deleted.

3.4 Definitions

3.4.1 General

3.4.1.1 **Error**, a deviation from the intended design which could result in unintended system behaviour or failure (CENELEC prEN 50129).

3.4.1.2 **Failure**, a deviation from the specified performance of a system. A failure is the consequence of a fault or error in the system (CENELEC prEN 50129).

3.4.1.3 **Fault**, an abnormal condition that could lead to an error in a system. A fault can be random or systematic (CENELEC prEN 50129).

3.4.1.4 **Time to failure detection**, is defined as the maximum time from the time point when an error or fault occurs until the failure is detected.

3.4.1.4.1 A **node** has one physical bus address. A node may have several logical addresses, Service Access Point, SAP.

3.4.1.5 A logical connection shall consist of one Logical connection Master and one logical connection Slave.

3.4.2 Logical connection Master

3.4.2.1 **Logical connection Master** is defined as the node sending the Connect Request Telegram for a point-to-point connection in the Safe Link Layer .

3.4.2.1.1 Logical connection Master is the master of a logical connection. A logical connection has one and only one master.

3.4.3 Logical connection Slave

3.4.3.1 **Logical connection Slave** of point-to-point connection is defined as the node sending the Connect Confirm telegram in the Safe Link Layer.

3.4.3.1.1 Logical connection Slave is the slave of a logical connection. A point-to-point logical connection has one and only one Logical connection Slave.

3.4.4 Multicast

3.4.4.1 A Multicast connection has one **Multicast Sender**.

3.4.4.2 A Multicast connection may have zero, one or several **Multicast Receivers**.

3.4.5 Reference Clock

3.4.5.1 There shall be one and only one node with the function **Reference Clock**. This function shall include a free running physical clock as a time base and a function to transmit the value of this clock to the bus.

3.4.5.1.1 This node with the function Reference Clock is defined as the **Reference Clock node**.

3.4.5.2 The **Reference_Clock_Time** is defined as the value of the time base (free running physical clock) inside the Reference Clock function.

3.4.5.2.1 Note: The Reference_Clock_Time is a digital number and it is not related to universal time.

3.4.5.2.2 Note: The Reference_Clock_Time is more advanced (higher value) than the Reference Time, due to process and transmission delays.

3.4.6 Reference Time

3.4.6.1 **Reference Time (RefTime)** is defined by the Sync And Reference Time Multicast messages on the bus and the moment of their transmission on the bus.

3.4.6.1.1 The Reference Time shall be the time common to all nodes on the bus.

3.4.7 Local Clock

3.4.7.1 Each bus node shall have a function called **Local Clock**. The Local Clock shall include a free running physical clock as a time base and a function for synchronisation.

3.4.7.1.1 A node with the function Local Clock is defined as **Local Clock Node**.

3.4.7.1.2 Exception: The node including the Reference Clock may not have a Local Clock function.

3.4.7.2 **Local_Clock_Time** is defined as the value of the time base inside the Local Clock, i.e. the value of the free running “physical” clock. The Local_Clock_Time as such is not synchronised with any other clock.

3.4.7.3 **Local Clock Inaccuracy (LCI)** is defined as the inaccuracy of the physical clock in a node in relation to an ideal clock due to hardware architectural constraints (includes clock resolution, clock drift, ...).

3.4.8 Local Reference Time

3.4.8.1 **Local_Reference_Time** is defined as the estimation of Reference Time by Local Clock based on the Local_Clock_Time and an offset determined by a synchronisation function inside the Local Clock.

3.4.8.1.1 This monitored time is used for time stamping and represents a system-wide reference time.

3.4.8.1.2 Exception: The node including the Reference Clock may use a fixed offset to the Reference_Clock_Time for estimation of Reference Time.

3.4.9 STL_TIME_STAMP

3.4.9.1 STL_TIME_STAMP is defined as the Time stamp added to the message before transmission. The value of the time stamp shall be the Local_Reference_Time.

3.4.10 Transfer Times

3.4.10.1 **Sender_Static_Transfer_Time** is defined as the static error in the time stamp at transmission time. It shall be the minimum value of the ageing of timestamp that always occur before message is transmitted.

3.4.10.1.1 Note: This ageing is due to the constant delay of Safe Link Layer and other computation that always occurs in conjunction with sending a message.

3.4.10.1.2 The delay caused by waiting for the send token according to the FDL protocol specified in /1/ shall not be included in this constant.

3.4.10.2 **Sender_Dynamic_Transfer_Time** is defined as the additional maximum estimated delay in the value of the ageing of time stamp at transmission time. This time shall include the sender LCI.

3.4.10.2.1 Note: The sender delay is not constant. The variation of this delay may be due to processing other tasks or handling interrupts.

3.4.10.2.2 The delay caused by waiting for the send token according to the FDL protocol specified in /1/ shall not be included in this constant.

3.4.10.2.3 Note: The minimum Transfer Time at the sender side is the value of Sender_Static_Transfer_Time and the maximum is Sender_Static_Transfer_Time + Sender_Dynamic_Transfer_Time.

3.4.10.3 Note: The Sender_Static_Transfer_Time and the Sender_Dynamic_Transfer_Time are used for the calculation of the maximum and minimum allowed age of a message, to ensure that a message which exceeds the time limit is rejected by the STL.

3.4.10.4 **Static_Bus_Transfer_Time** is defined as the value of the time to transfer the minimum frame (One byte of application data).

3.4.10.4.1 Note: This value depends on the baud rate of the bus.

3.4.10.5 **Dynamic_Bus_Transfer_Time** is defined as the value of the time to transfer twice the maximum frame (One re-transmission) plus Token Rotation Time according to /1/.

3.4.10.5.1 Note: This value depends on the baud rate of the bus.

3.4.10.6 The **Safety Time Tolerance** STT is defined as the tolerance of age at reception of safe messages.

3.4.10.7 **Receiver_Static_Transfer_Time** is defined as the static delay in the receiving node. It shall be the minimum delay that always occurs before the message is processed in the STL.

3.4.10.8 Note: This delay is due to the constant delay of Safe Link Layer and other computation that always occurs in conjunction with receiving a message.

3.4.10.9 **Receiver_Dynamic_Transfer_Time** is defined as the additional maximum estimated delay in the receiving node.

3.4.10.10 Note: The receiver delay is not constant. The variation of this delay may be due to processing other tasks or handling interrupts.

3.4.10.11 Note: The minimum **Transfer Time** at the receiver side is the value of Receiver_Static_Transfer_Time and the maximum is Receiver_Static_Transfer_Time + Receiver_Dynamic_Transfer_Time.

3.4.10.12 Note: The Receiver_Static_Transfer_Time and the Receiver_Dynamic_Transfer_Time are used for the calculation of the maximum and minimum allowed age of a message, to ensure that a message which exceeds the time limit is rejected by the STL.

4. SUMMARY DESCRIPTION

4.1.1.1 In the Safe Time Layer the main tasks are to monitor the local clocks of the two communicating bus nodes, and to monitor and control the age of the messages.

4.1.1.2 Through the network of nodes using the Safe Time Layer, all local clocks are monitoring each other, directly or indirectly.

4.1.1.3 Example for an STM architecture:

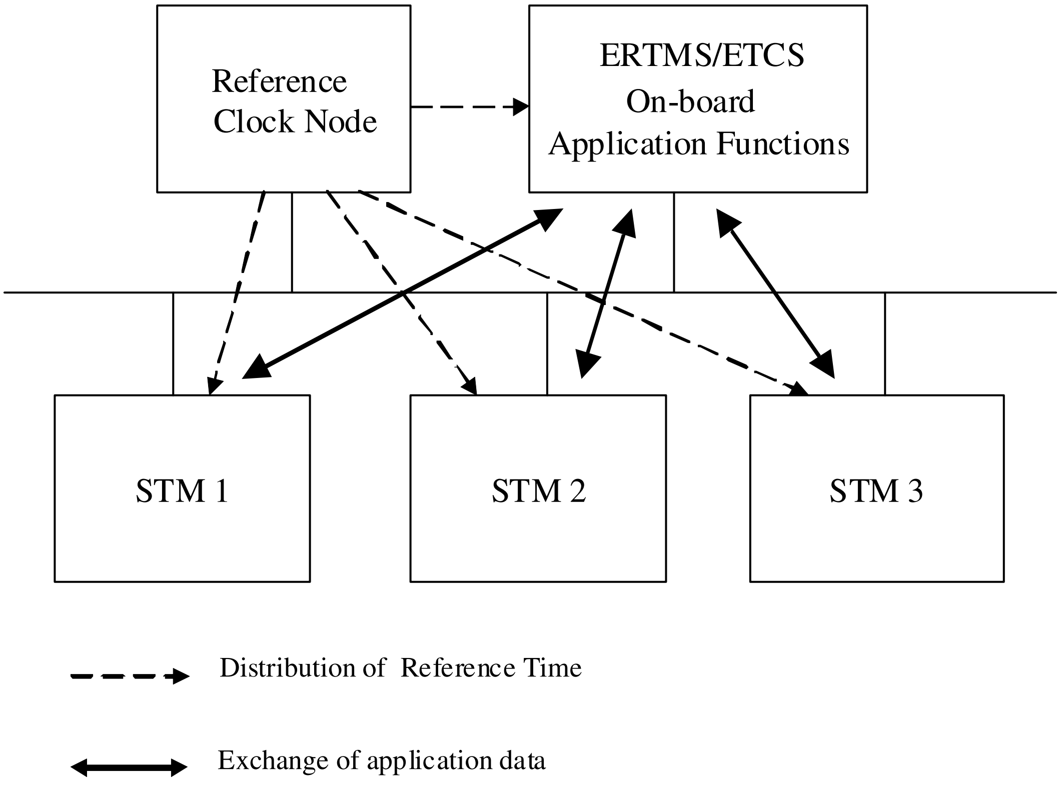
Reference Clock Node
ERTMS/ETCS On-board Application Functions
STM 1
STM 2
STM 3
Distribution of Reference Time
Exchange of application data

Figure 1 System architecture example with several STMs

4.1.1.4 Intentionally deleted

4.1.1.5 The STL_Time_Stamp shall be added to all sent application messages in order to monitor and control the ageing of the message.

4.1.1.6 The STL_Time_Stamp shall be added to the sent message as late as possible and has the value of Local_Reference_Time soon before transmission.

4.1.1.7 The receiving node shall check the received STL_Time_Stamp. Its age shall be within a certain window Safety Time Tolerance, STT (see Section 7.8).

4.1.1.8 Reference Time Transfer

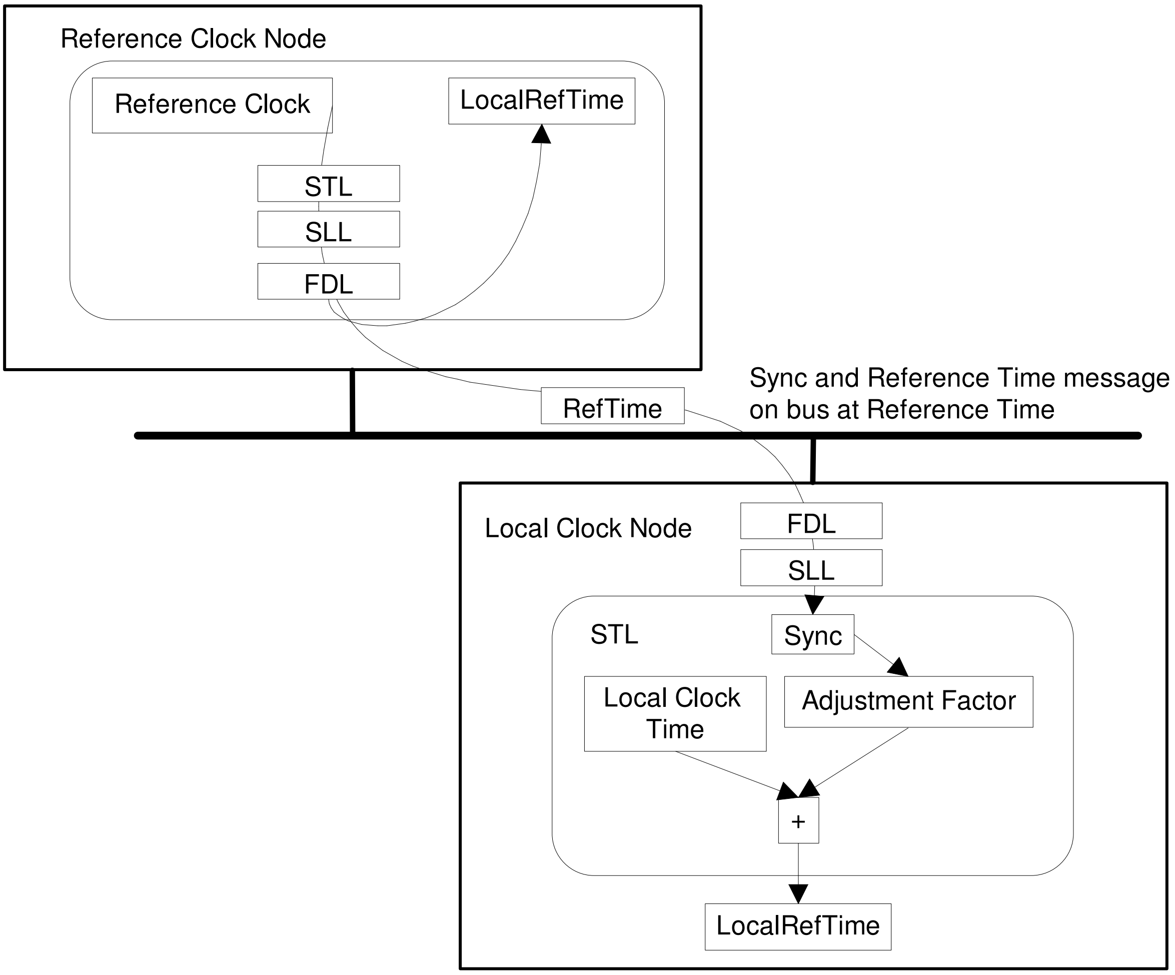

Figure 2 Reference Time transfer and processing

4.2 Time stamps

4.2.1.1 There are time stamps on two different levels. The time stamps on the Safe time layer (STL_Time_Stamp) keep track of the maximum time delay of data sent via the bus. The time stamps on the signalling/application layer are used to keep track on the ageing of the signalling/application data in a wider perspective.

4.2.1.2 Note: the time stamp functionality shown in the diagram on application level uses the STL layer Reference Time service.

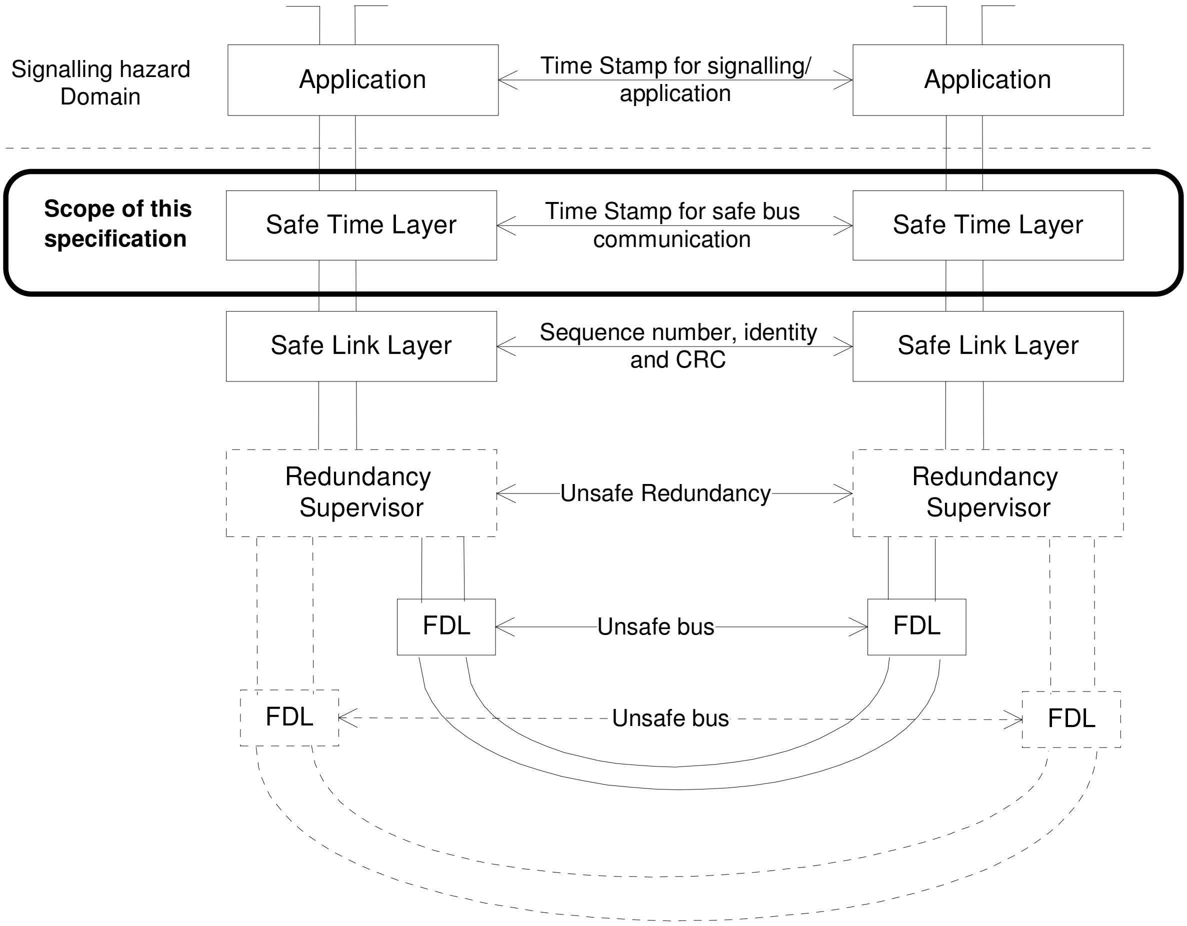

Figure 3 Scope of the protocol specification

5. SAFE TIME LAYER CONTROL

5.1 Start-up of Safe Time Layer

5.1.1.1 The Reference Clock Node shall start to multicast Sync and Reference Time messages as soon as possible after the node has been powered on. This provides Reference Time (RefTime) to all other nodes.

5.1.1.2 Local_Reference_Time shall be derived from the Local_Clock_Time and the received Reference Time by the Local Clock Node.

5.1.1.3 During rest of session, the Local_Reference_Time shall be checked and adjusted against the Reference Time.

5.1.1.4 The Logical connection Slave and the Logical connection Master shall be initiated only after the Local_Reference_Time is synchronised relative to the Reference Time. This shall be done by issuing Ready to Run message responded by a Run message (Ready to Run and Run messages shall be issued only after the corresponding connection is established by the Safe Link Layer).

5.1.1.5 An example of a successful start-up sequence of the Safe Time Layer is shown in the figures 'Figure 4 Safe Time Layer start up example: system architecture' and 'Figure 5 Safe Time Layer start up example: sequence chart'.

5.1.1.6

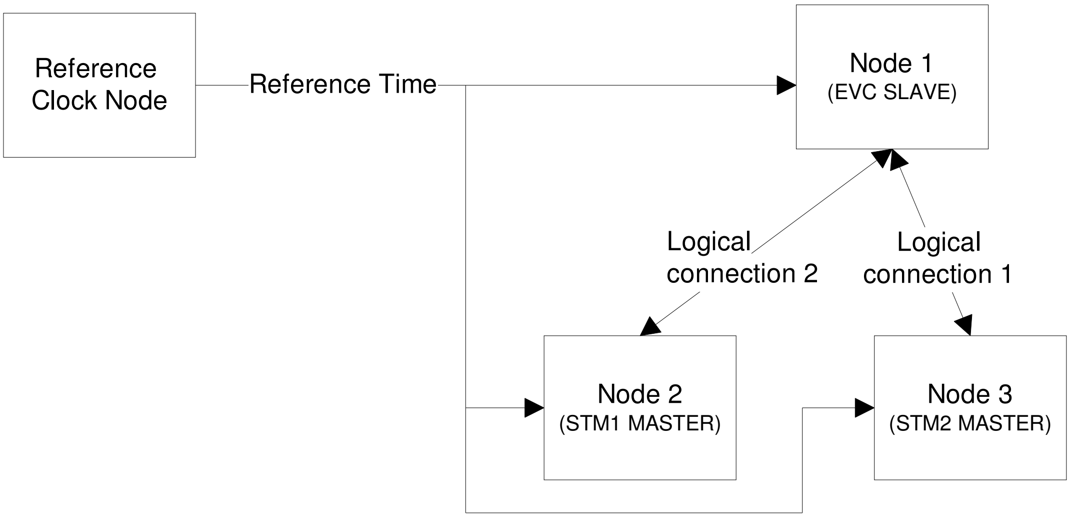

Figure 4 Safe Time Layer start up example: system architecture

5.1.1.7

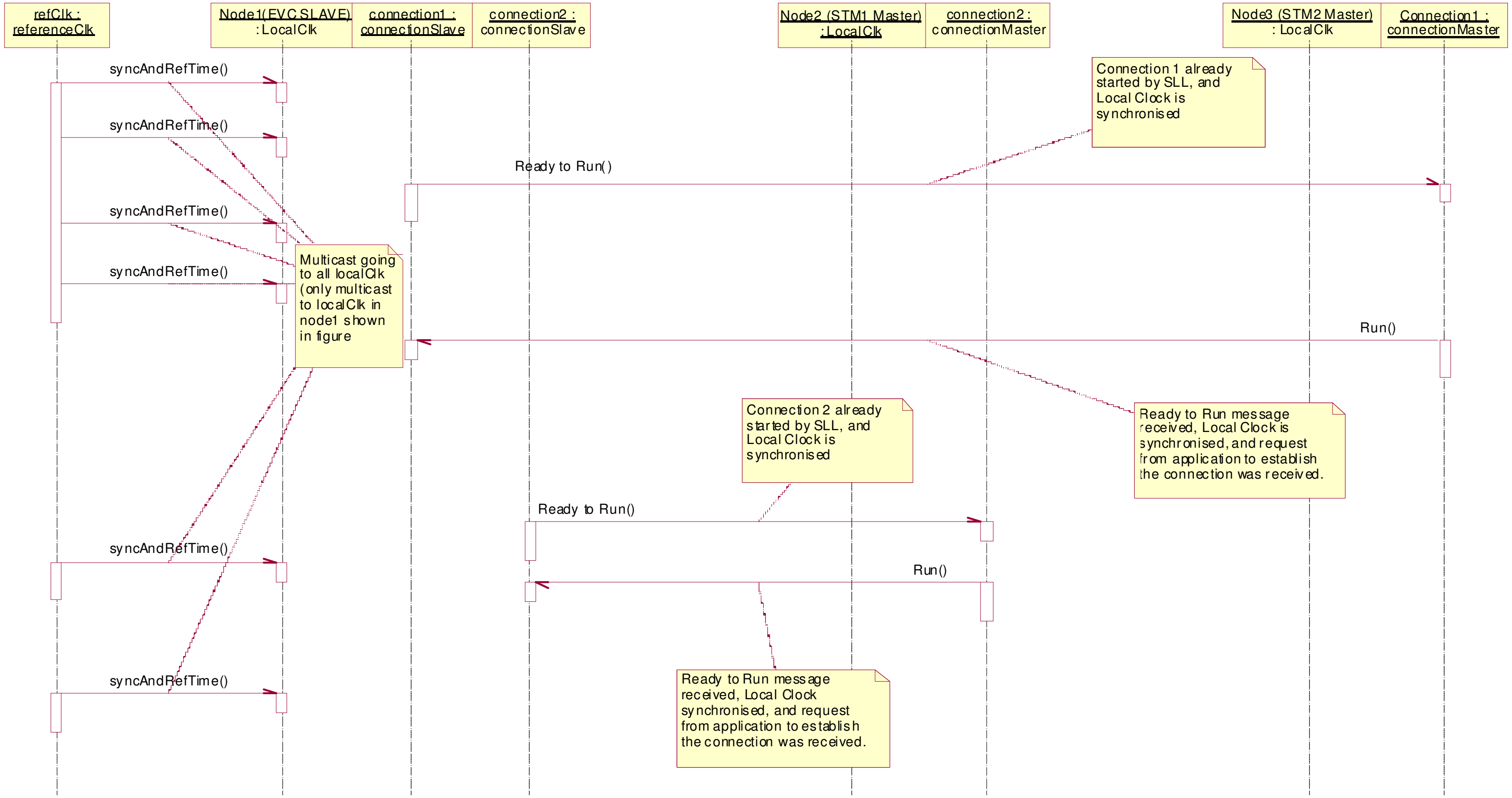
refClk :
referenceClk
Node1(EVC SLAVE)
: LocalClk
connection1 :
connectionSlave
connection2 :
connectionSlave
Node2 (STM1 Master)
: LocalClk
connection2 :
connectionMaster
Node3 (STM2 Master)
: LocalClk
Connection1 :
connectionMaster
syncAndRefTime()
syncAndRefTime()
syncAndRefTime()
syncAndRefTime()
Connection 1 already started by SLL, and Local Clock is synchronised
Ready to Run()
Multicast going to all localClk (only multicast to localClk in node1 shown in figure
Run()
Connection 2 already started by SLL, and Local Clock is synchronised
Ready to Run message received, Local Clock is synchronised, and request from application to establish the connection was received.
Ready to Run()
syncAndRefTime()
Run()
syncAndRefTime()
Ready to Run message received, Local Clock synchronised, and request from application to establish the connection was received.

Figure 5 Safe Time Layer start up example: sequence chart

5.2 Configuration data of Safe Time Layer

5.2.1.1 Connect Request Telegram and Connect Confirm Telegrams in the Safe Link Layer shall include fields for configuration data of the Safe Time Layer.

5.2.1.2 The Configuration Data Field is specified in the table below:

5.2.1.3

Variable	Length	Comment
Configuration data prefix X	8	value = 3.
Configuration data prefix Y	8	value = 0.
Configuration data prefix Z	8	value = 0.
Sender_Dynamic_Transfer_Time	32	The Sender_Dynamic_Transfer_Time of the sender (signed integer, complement to two code) [ms].
Sender_Static_Transfer_Time	32	The Sender_Static_Transfer_Time of the sender (signed integer, complement to two code) [ms].

5.3 Disconnect Reason and Type

5.3.1.1 Detected failures in the Safe Time Layer shall result in issuing a Disconnect request to the Safe Link Layer.

5.3.1.2 The disconnect_reason and disconnect_type for the Disconnect request to Safe Link Layer issued by the Safe Time Layer shall be set according to the table below.

5.3.1.2.1 Note: After a Disconnect request with disconnect_type final sent, the connection state changes to 'Disconnect Final' and cannot be established again, see chapters 7.6 and 7.7. Also the SLL does not allow reestablishment of the connection in that case.

5.3.1.3

#disconnect _reason	Disconnect_ type	Description
20h		Not used (reserved for future use)
21h	Non-Final	Local Clock is not yet synchronised
22h	Final	Local Clock; lost synchronisation see chapter 7.4.1.5
23h	Non-Final	Logical Connection Master; time-out at connection start-up
24h	Non-Final	Logical Connection Slave; time-out at connection start-up
25h	Non-Final	STL_Time_Stamp of a received telegram does not fulfill the STTmin criterion
26h	Non-Final	STL_Time_Stamp of a received telegram does not fulfill the STTmax criterion
27h-3Eh		Not used (reserved for future use)
3Fh		Other (not in this list) disconnect_reason of Safe Time Layer
40h	Final or Non-Final	Disconnect on request by application

5.4 STL_Time_Stamp

5.4.1.1 The STL_Time_Stamp shall be a 32 bit time stamp added at the end of the message.

5.4.1.2 Resolution shall be 1 ms.

5.4.1.2.1 Note: The resolution and the number of bits lead to a time to wrap around of 4294967296 ms, which is every 49 days, 17 hours, 2 minutes, 47 seconds and 296 milliseconds.

5.4.1.3 All time stamps transferred over the bus shall be Local_Reference_Time or RefTime.

6. MESSAGES

6.1.1.1 The format of the data added by the Safe Time Layer shall be the little endian format low-order byte first; i.e. 16- and 32-bit values for example are transferred bytewise, and the bytes are transferred as follows:

16-bit values: lowbyte, highbyte

32-bit values: lowword.lowbyte, lowword.highbyte, highword.lowbyte, highword.highbyte

6.1.1.2 The table below specifies the command numbers that shall be used for the messages of the Safe Time Layer.

6.1.1.3

Type	Value	Message
SL-4 messages	89h	Application Data (point to point)
	8Dh	Application Data (multicast)
	A1h	Sync and Reference Time
	A2h	Ready to Run
	A3h	Run
	A4h	Safe Time Layer Startup
	A0h, A5h..BFh	Reserved for future extension
SL-2 messages	09h	Application Data
	22h	Ready to Run
	23h	Run
	20h, 21h, 24h..3Fh	Reserved for future extension
SL-0 messages	C9h	Application Data
	E2h	Ready to Run
	E3h	Run
	E0h,E1h,E4h..FFh	Reserved for future extension

6.1.1.4 These commands numbers shall be passed to the lower layer (e.g. SLL).

6.1.1.5 Note: The Safe Link Layer handles all these Safe Time Layer telegrams as data telegrams. In the following chapters the content of each telegram is specified as Safe Link Layer net_data. The Safe Time Layer and the Safe Link Layer use a common field for the command number as described in the data telegram structure in reference /2/.

6.1.1.6 Telegrams with command numbers reserved for future extension shall be ignored by the receiver.

6.2 Application Data

6.2.1.1 Application data shall be used by the application layer to transmit multicast data telegrams or point-to-point data telegrams.

6.2.1.2

Description	Application Data		
Transmission media	STM FFFIS		
Content	**Variable**	**Length**	**Comment**
	Application data	N*8	ERTMS language packets etc.
	STL_Time_Stamp	32	Time stamp of the message with the Local_Reference_Time (unsigned integer) [ms].

6.3 Sync and Reference Time

6.3.1.1 The Sync and Reference Time messages shall be sent by the Reference Clock using Multicast.

6.3.1.2 The transmission of the Sync and Reference Time messages shall start as soon as possible after the node has been powered on.

6.3.1.3 The Reference sync number shall be a counter value that increments by one for every sent message of the Sync and Reference Time message.

6.3.1.4 The initial value for Reference Sync number shall be 0h.

6.3.1.5 The Reference sync number shall not wrap around.

6.3.1.5.1 Note: With a cycle time of 250ms a wrap around is reached after 24 years and 17 days.

6.3.1.6 The time of sending a Sync and Reference Time message on the bus shall be captured by the Reference Clock Node as accurately as possible. This captured time of sending shall define the Reference Time when the message was on the bus.

6.3.1.7 The captured time shall be distributed to all other nodes with the following Sync and Reference Time message as the parameter Reference Time [n-1].

6.3.1.8 The Reference Time of the first message sent from the reference clock shall be set to zero. The local clocks shall not use the Reference Time value of this first message.

6.3.1.8.1 Note: This is because the Reference Time parameter does not refer to a previous Sync and Reference Time message, as it is the first one.

6.3.1.9

Description	Sync and Reference Time		
Transmission media	STM FFFIS		
Content	**Variable**	**Length**	**Comment**
	Configuration data prefix X	8	value = 3
	Configuration data prefix Y	8	value = 0
	Configuration data prefix Z	8	value = 0
	Reference Sync [n]	32	The messages number. (unsigned integer).
	Reference Time [n-1]	32	Captured time of previous message see 6.3.1.6 and see also 7.2 (unsigned integer) [ms].

6.4 Ready to Run

6.4.1.1 The Ready to Run message shall be a point-to-point message.

6.4.1.2 The Ready to Run message shall be sent by the logical connection slave and only when the Local Clock is synchronised.

6.4.1.3 The Ready to Run message shall only be sent when a new connection is established by the Safe Link Layer.

6.4.1.4

Description	Ready to Run		
Transmission media	STM FFFIS		
Content	**Variable**	**Length**	**Comment**
	STL_Time_Stamp	32	Time stamp of the message with the Local_Reference_Time. (unsigned integer) [ms].

6.5 Run

6.5.1.1 The Run message shall be a point-to-point message.

6.5.1.2 The Run message shall be sent by a logical connection master when the Local Clock is synchronised and the logical connection master has got the Ready to Run message from the logical connection slave.

6.5.1.3 No Application Data messages shall be sent by the Logical Connection Master before a Run message has been sent via the logical connection.

6.5.1.4 No Application Data messages shall be sent by the Logical Connection Slave before a Run message has been received by the logical connection.

6.5.1.5

Description	Run		
Transmission media	STM FFFIS		
Content	**Variable**	**Length**	**Comment**
	STL_Time_Stamp	32	Time stamp of the message with the Local_Reference_Time. (unsigned integer) [ms].

6.6 Safe Time Layer Startup for multicast

6.6.1.1 The Safe Time Layer Startup messages shall be sent by all multicast senders using Multicast, except for the Reference Clock Function.

6.6.1.1.1 Note: For Reference Clock Function there is a specific message for time synchronisation (see chapter 6.3).

6.6.1.2 The Safe Time Layer Start-up message shall be sent cyclically during start up time. This message shall also be sent after start up time to allow receivers to get the message when they are ready for it (they have started up).

6.6.1.2.1 No multicast application messages shall be sent before the local clock is synchronised and no Multicast application messages shall be processed by the receiver before the local clock is synchronised.

6.6.1.2.2 No multicast application messages shall be sent before a STL Start-up message has been sent and no Multicast application messages shall be processed by the receiver before it has received a STL Start-up message.

6.6.1.3 The interval for the messages shall be inside SafeTimeLayerStartupInterval (see chapter 9.3). This is applicable until StartupSynchronisationTimeLimit (see chapter 9.5).

6.6.1.4 After expiration of StartupSynchronisationTimeLimit the interval shall be inside SafeTimeLayerReStartInterval (see chapter 9.5).

6.6.1.5

Description	Safe Time Layer Startup		
Transmission media	STM FFFIS		
Content	**Variable**	**Length**	**Comment**
	Configuration data prefix X	8	value = 3
	Configuration data prefix Y	8	value = 0
	Configuration data prefix Z	8	value = 0
	Sender_Dynamic_Transfer_Time	32	The Sender_Dynamic_Transfer_Time of the sender (signed integer, complement to two code) [ms].
	Sender_Static_Transfer_Time	32	The Sender_Static_Transfer_Time of the sender (signed integer, complement to two code) [ms].

7. SPECIFICATION OF FUNCTIONS

7.1 GENERAL

7.1.1.1 'Figure 6 Message transfer times' illustrates time stamping of a transferred message performed by the sender and receiver.

7.1.1.2

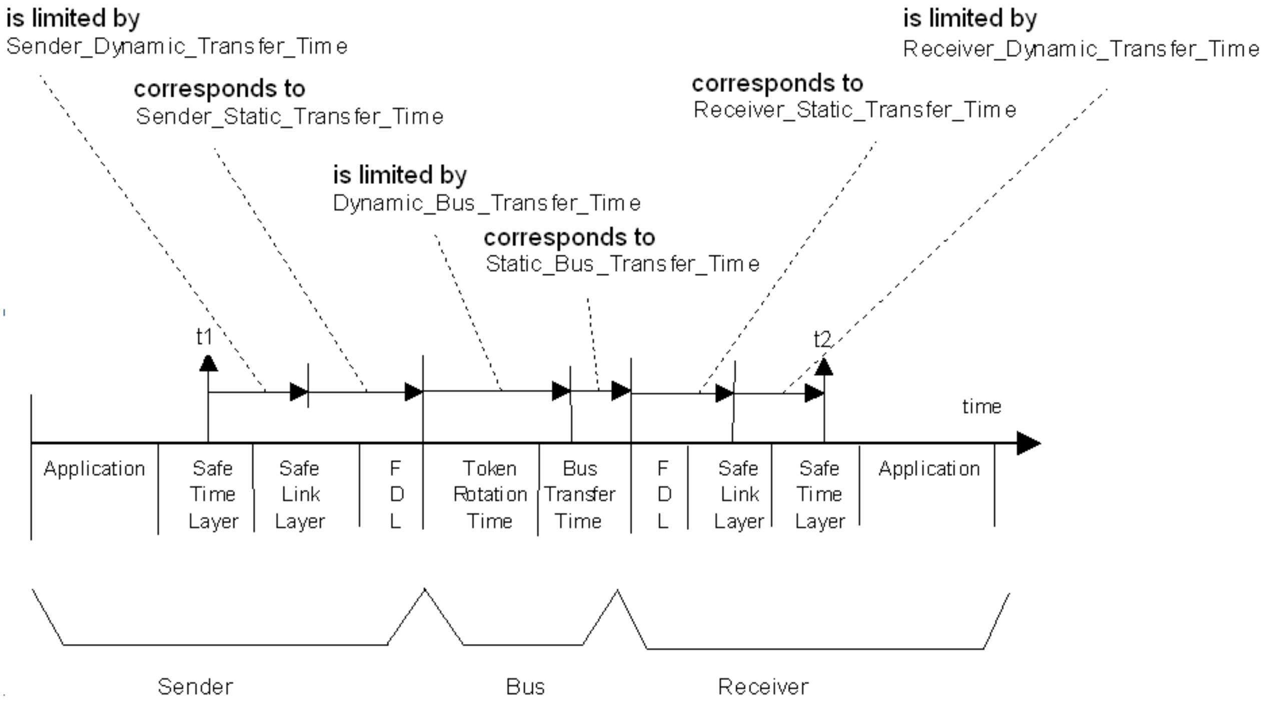

Figure 6 Message transfer times

7.1.1.2.1 t1 is STL_Time_Stamp (or RefTime[n-1] in case of a Sync and Reference Time message).

7.1.1.2.2 t2 is Local_Clock_Time (or LocalTime[n-1] in case of a Sync and Reference Time message).

7.1.1.2.3 The time stamping itself (on the senders side) is not a safety function. However the monitoring of the time stamp (on receivers side) is part of the safety process.

7.1.1.2.4 Intentionally deleted

7.1.1.3 Wrap around is allowed for any clock value or time stamp.

7.1.1.3.1 Justification: Wrap around in time stamping can occur every 2^{32} ms, and message chronology is guaranteed by the sequence number.

7.2 Principle of Clock Synchronisation

7.2.1.1 In order to solve the hazard of ‘missing time deadlines of safety critical data transferred between physical nodes’ some mechanism to synchronise the clocks or translate timestamps from one clock to another clock must be introduced.

7.2.1.2 The principle is translation of timestamps from local time bases to a common time base.

7.2.1.3 Synchronisation also takes care of clock drift, defined as a difference between Local_Clock_Time and Reference Time over a period of time.

7.2.1.4 The clock drift between the clocks is supervised. Low rates of drift are compensated. High rates of drift are detected as failure and lead to disconnection.

7.2.1.5 The Reference Clock cyclically multicasts the Reference Time to the bus.

7.2.1.6 All Local Clocks cyclically calculate the difference between the Local_Clock_Time and the received Reference Time (RefTime) by determining the Adjustment Factor. See 7.4.4 Calculation of Adjustment Factor.

7.2.1.7 All Local Clocks supervise the clock drift between Local_Clock_Time and the Reference Time. This is done according to section 7.4.3 Resynchronise.

7.2.1.7.1 If the Local Clock finds the clock drift out of specified tolerance the node isolates itself. This is done according to section 7.4.1.

7.2.1.8 Intentionally deleted.

7.2.1.9 All nodes shall fulfil the specified timing requirements in reference /3/.

7.2.1.10 The Figure 7 Reference Time transmission by Reference Clock Node illustrates how the Reference Clock Node transfers the Reference Time to a Local Clock Node.

7.2.1.11

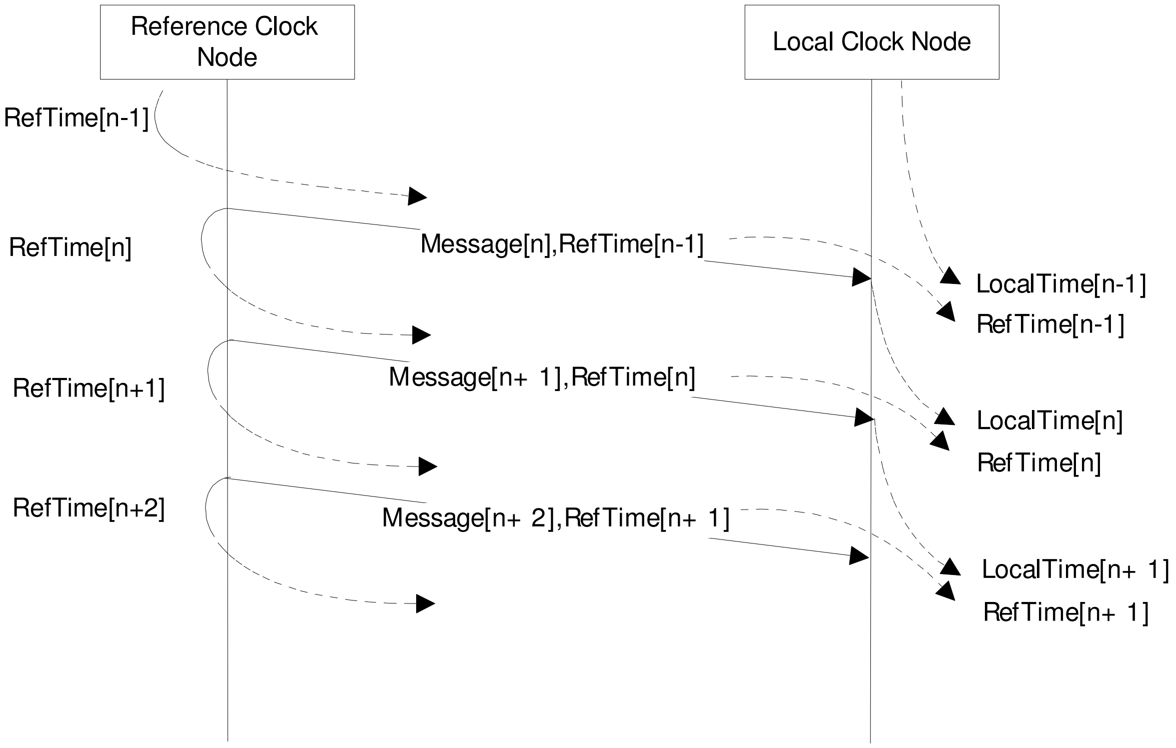

Figure 7 Reference Time transmission by Reference Clock Node

7.2.1.12 The Reference Clock shall save the Reference_Clock_Time as RefTime[n] when sending the Message[n].

7.2.1.13 The Local Clock shall save the Local_Clock_Time as LocalTime[n] at reception of Message[n].

7.2.1.14 The Reference Clock shall include RefTime[n] in Message[n+1].

7.2.1.15 The Figure 7 shows that the Local Clock Node has got the Reference_Clock_Time value RefTime[n] and its own value LocalTime[n] for the common event, transferring of Message[n], after a delay of one message.

7.2.1.16 The Local clock is now able to calculate the time difference between the Reference_Clock_Time and the Local_Clock_Time.

7.2.1.17 Since the Reference Clock cyclically sends its time, the Local Clock is able to supervise the difference history. The clock drift supervision is one of them.

7.2.1.18 Note: The method allows to minimise the error in delay on the sender side by capturing the RefTime[n] as close as possible to the time of transmission (after the CRC is calculated).

7.2.1.19 Note: For maximum accuracy of the Reference Clock, the difference between the stored time and the time the telegram is transmitted on the bus should vary as less as possible. If measuring time of transmission from the bus controller, the error introduced by waiting for the token according to /1/ can be eliminated.

7.2.1.20 **Note on possible safety risk:** The specified clock synchronisation principle does not directly detect a constant offset, existing from start, between the Reference Time on the bus and the Local Reference Time in the clock node if the clock node does not implement a Local Clock. However no point-to-point connection setup will be possible, if such an offset exists, because the telegram age check for the received Run or Ready to Run telegram received in the clock node will fail. Thus an ETCS onboard function in the clock node corrupted by this problem will stay unavailable. If the clock node also implements the odometer function additional measures are necessary as e.g. to start the odometer function only when a first point-to-point connection with a safe Local Clock node has been successfully established.

7.3 Reference Clock

7.3.1.1 **The Referenc**e Clock shall start to send Sync and Reference Time (multicast) messages as soon as possible after the node has been powered on.

7.3.1.1.1 Note: This is independent from the initialisation of the point to point connections by the Safe Link Layer.

7.3.1.2 The Reference Clock shall cyclically send Sync And Reference Time messages.

7.3.1.3 During synchronisation of the Safe Time Layer the Reference Clock shall send Sync and Reference Time Messages with a short interval.

7.3.1.4 The interval for the messages shall be inside SyncAndRefTimeSyncInterval (see chapter 9.9). This is applicable until SyncAndRefTimeStartupTimeLimit (see chapter 9.11).

7.3.1.5 After expiration of SyncAndRefTimeStartupTimeLimit the interval becomes longer, and shall be inside SyncAndRefTimeRunInterval (see chapter 9.10).

7.3.1.6 The state machine in Figure 8 specifies the requested behaviour of the Reference Clock.

7.3.1.7

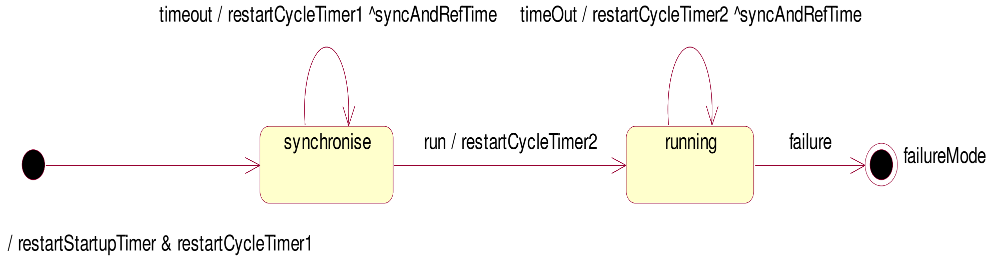
timeout / restartCycleTimer1 ^syncAndRefTime
timeOut / restartCycleTimer2 ^syncAndRefTime
synchronise
run / restartCycleTimer2
running
failure
failureMode
/ restartStartupTimer & restartCycleTimer1

Figure 8 Reference Clock state machine

7.3.1.7.1 States shall be:

7.3.1.7.1.1 Synchronise (state): send Sync and Reference Time messages with a short interval.

7.3.1.7.1.2 Running (state): send Sync and Reference Time messages with a long interval.

7.3.1.7.1.3 FailureMode (state): stop sending SyncAndReftime. Receivers will detect the failureMode after timeout.

7.3.1.7.2 Events shall be:

7.3.1.7.2.1 TimeOut (event): expiration of the Cycle Timer.

7.3.1.7.2.2 Run (event): expiration of the Startup Timer.

7.3.1.7.2.3 Failure (event): the event of failure in the RefTime node.

7.3.1.7.3 Actions shall be:

7.3.1.7.3.1 RestartStartupTimer (action): set a timer to SyncAndRefTimeStartupTimeLimit .

7.3.1.7.3.2 restartCycleTimer1 (action): set a timer to a value inside SyncAndRefTimeSyncInterval.

7.3.1.7.3.3 restartCycleTimer2 (action): set a timer to a value inside SyncAndRefTimeRunInterval.

7.3.1.7.4 Send-clauses shall be:

7.3.1.7.4.1 SyncAndRefTime (send-clause): multicast a Sync and Reference Time Message.

7.4 Local Clock

7.4.1 General

7.4.1.1 The Local Clock shall not provide Local_Reference_Time before it is synchronised.

7.4.1.2 Local Clocks shall check that its drift compared to Reference Time is not outside limits.

7.4.1.3 A Local Clock shall calculate the Adjustment Factor and supervise the difference in time between Local_Clock_Time and Reference Time.

7.4.1.4 If a node contains the Reference Clock, a Local Clock is not required on the node.

7.4.1.5 A Local Clock shall be considered as synchronised only as long as:

7.4.1.5.1 an adjustment factor can be calculated, see chapter 7.4.4

7.4.1.5.2 and Reference Time is a monotonic increasing value, see chapter 7.4.3.3

7.4.1.5.3 and Local_Clock_Time is a monotonic increasing value, see chapter 7.4.3.4

7.4.1.5.4 and the short term drift between the local clock and the reference clock is inside limits, see chapter 7.4.3.5

7.4.1.5.5 and a valid number of Sync And Reference Time messages have been received, see chapter 7.4.3.6

7.4.1.5.6 and the long term drift between the local clock and the reference clock is inside limits, see chapter 7.4.3.7

7.4.1.5.7 and a resynchronisation is performed within a specified time, see chapter 7.4.1.6.

7.4.1.6 The maximum time interval between two successfully executed resynchronisations (see chapter 7.4.3) shall be limited.

7.4.1.6.1 Maximum time between two successfully executed resynchronisations shall be localClockMaxReSyncInterval (see chapter 9.2).

7.4.1.7 The state machine in Figure 9 specifies the requested behaviour of a Local Clock:

7.4.1.7.1

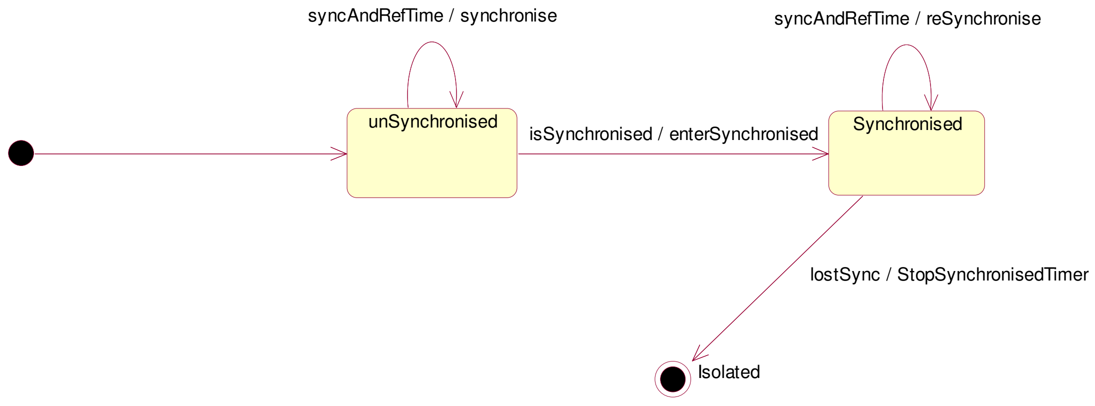
syncAndRefTime / synchronise
syncAndRefTime / reSynchronise
unSynchronised
isSynchronised / enterSynchronised
Synchronised
lostSync / StopSynchronisedTimer
Isolated

Figure 9 Local Clock state machine

7.4.1.7.2 States shall be:

7.4.1.7.2.1 unSynchronised (state): Local Clock is not synchronised.

7.4.1.7.2.2 Synchronised (state): Local Clock is synchronised.

7.4.1.7.2.3 Isolated (state): failure state, Local Clock is not synchronised.

7.4.1.7.3 Events shall be:

7.4.1.7.3.1 SyncAndRefTime (event): reception of a Sync and Reference Time Message from the Reference Clock.

7.4.1.7.3.2 IsSyncronised (event): this event occurs when the Local Clock is synchronised.

7.4.1.7.3.3 LostSync (event): event of failure in synchronisation that occur when the Synchronised Timer expires, or any of the conditions specified in 7.4.3 fail.

7.4.1.7.4 Actions shall be:

7.4.1.7.4.1 Synchronise (action): performs synchronising of the Local Clock, specified in Section 7.4.2.

7.4.1.7.4.2 EnterSynchronised (action): Reset a timer called Synchronised Timer.

7.4.1.7.4.3 ReSyncronise (action): performs resynchronising of the Local Clock, specified in Section 7.4.3 and restart the Synchronised Timer.

7.4.1.7.4.4 StopSyncronisedTimer (action): stop the Synchronised Timer and disconnect all logical connections on the node, including multicast (isolated).

7.4.2 Synchronise

7.4.2.1 The Local Clock function by action Synchronise shall calculate the first valid Adjustment factor (see chapter 7.4.4).

7.4.2.2 The following actions shall be performed at reception of each valid Sync and Reference Time Message (Message[n+1] in Figure 7) except for the first received Sync and Reference Time Message. If one of the actions fails the Local clock shall be kept unsynchronised.

7.4.2.2.1 Check that refTime[n] > refTime[n-1]

7.4.2.2.2 Check that localTime[n] > localTime[n-1]

7.4.3 Resynchronise

7.4.3.1 The Local Clock function by action Resynchronise shall recalculate the actual adjustment factor (see chapter 7.4.4)

7.4.3.2 The following actions shall be performed at reception of a valid Sync and Reference Time Message (Message[n+1] in Figure 7). If one of the actions fails the Local clock shall get unsynchronised (event lostSync) and Local_Reference_Time shall be undefined.

7.4.3.3 Check that refTime[n] > refTime[n-1]

7.4.3.4 Check that localTime[n] > localTime[n-1]

7.4.3.5 Check that |refTime[n] – localTime[n] – adjustmentFactor| < MaxClockInaccuracyAfterAdjustFactor (see chapter 9.6).

7.4.3.6 Check that the number of valid received sync and reference time messages is at least MinNumberofSyncAndRefMsgReceived (see chapter 9.8) during a TimeForLongTermDriftCheck (see chapter 9.7) time frame. The time frame shall not be floating, i.e. the start and end times of the time frame shall be set at one point and not changed until the end time has been passed.

7.4.3.7 Check that the long term drift between the local clock and the reference clock is within 0.1% by the following comparison:

```
0 ≤ |adjustmentFactor[n] - adjustmentFactor[k]| / TimeForLongTermDriftCheck
  ≤ 0.001
```

7.4.3.7.1 The long term drift shall be checked over a non-floating period specified by the TimeForLongTermDriftCheck time frame. The index value “n” in the above formula shall correspond to the Adjustment Factor which is valid at the end of the period (the last received Sync And Reference Time message used for calculation had the message number “n”). The index value ”k” in the above formula corresponds to the Adjustment Factor which is valid at the beginning of the period (the last received Sync And Reference Time message used for calculation had the message number “k”).

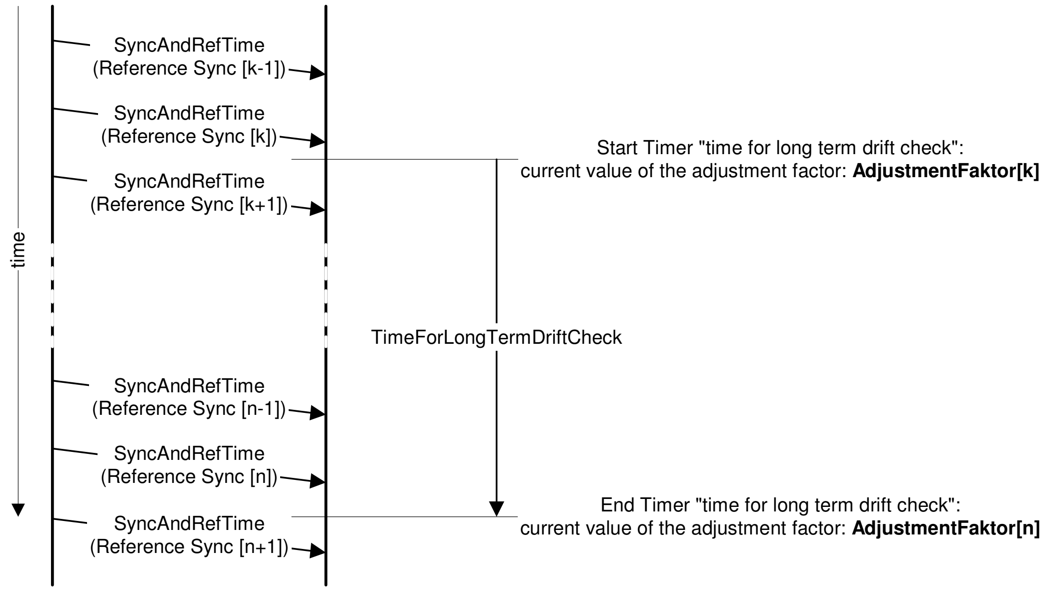
time
SyncAndRefTime (Reference Sync [k-1])
SyncAndRefTime (Reference Sync [k])
SyncAndRefTime (Reference Sync [k+1])
Start Timer "time for long term drift check": current value of the adjustment factor: **AdjustmentFaktor[k]**
TimeForLongTermDriftCheck
SyncAndRefTime (Reference Sync [n-1])
SyncAndRefTime (Reference Sync [n])
SyncAndRefTime (Reference Sync [n+1])
End Timer "time for long term drift check": current value of the adjustment factor: **AdjustmentFaktor[n]**

Figure 10 Long Term Drift check

7.4.3.7.2 Intentionally deleted

7.4.3.7.2.1 Note: The time to failure detection is therefore 2*TimeForLongTermDriftCheck.

7.4.3.7.2.2 Note: The intermediate drift within one check interval may be higher than the allowed long term drift. It is limited by two factors. The first factor is the short term drift check according to chapter 7.4.3.5. The second factor is the long term drift check itself that must be passed at end of each long term drift check interval.

7.4.4 Calculation of Adjustment Factor

7.4.4.1.1 The Adjustment Factor shall be calculated as the difference between the Local_Clock_Time recorded by Local Clock and the Reference Time from the corresponding valid telegram. The adjustment value shall be calculated as the running mean value.

7.4.4.2 The Adjustment Factor shall be calculated as:

$$\texttt{AdjustmentFactor} \leftarrow \frac{1}{16}\sum_{16} \texttt{(refTime[N] - localTime[N])}$$

7.4.4.2.1 If a Sync and Reference Time Message is lost on the bus the next one shall be used in the formula.

7.4.4.2.2 An Adjustment factor shall not be calculated for less than 16 pairs of refTime and localTime values.

7.4.5 Calculation of Local_Reference_Time

7.4.5.1 Calculation in Local Clock node: The Local_Reference_Time shall be the adjusted Local_Clock_Time to estimate the Reference Time on the bus. The Adjustment Factor shall be used to make that adjustment:

Local_Reference_Time ← AdjustmentFactor + Local_Clock_Time

7.4.5.2 Calculation in Clock node without Local Clock: The Local_Reference_Time shall be either the Reference_Clock_Time or optionally the corrected Reference_Clock_Time to estimate the Reference Time on the bus.

7.4.5.2.1 Direct calculation of Local_Reference_Time in Clock node without Local Clock:

Local_Reference_Time ← Reference_Clock_Time

Note: This variant may be used, if the ClockNodeAdjustmentFactor specified below is small and the resulting inaccuracy not relevant.

7.4.5.2.2 Corrected calculation of Local_Reference_Time in Clock node without Local Clock: The ClockNodeAdjustmentFactor shall be used to make the following adjustment using the estimated times in the table below.

Term	Description
Clock_Sender_Transfer_Time	Typical time from storing the Clock_Reference_Time for a Sync And Reference Time telegram until the telegram is ready for transmission. Note: Depending on when the time is stored, this time may be negative.
Bus_Transfer_Time	Typical time to wait for token and send the telegram.
Clock_Receiver_Transfer_Time	Typical time to receive a telegram. Note: This is only a virtual time, but including this time makes calculation in the clock node most similar to calculation in a Local Clock Node.

ClockNodeAdjustmentFactor ← - (Clock_Sender_Transfer_Time + Bus_Transfer_Time + Clock_Receiver_Transfer_Time)

Note: In general this time will be negative; because of the delay for transmission, the Clock Reference Time is ahead of the Reference Time on the bus.

Local_Reference_Time ← ClockNodeAdjustmentFactor + Reference_Clock_Time

7.5 Principle of Logical Connection

7.5.1 General for Point-to-point connection

7.5.1.1 A logical connection shall be used for transferring safety related application data.

7.5.1.2 A logical connection shall be unique by its SAP and physical addresses.

7.6 Logical Connection Master

7.6.1.1 This chapter specifies the requested behaviour of a Logical connection Master at the interface between the connected nodes:

7.6.1.1.1 Note: Every logical connection needs a separate state machine.

7.6.1.2 If the Master Local Clock is not synchronised no Connection request shall be issued to the Safe Link Layer.

7.6.1.3

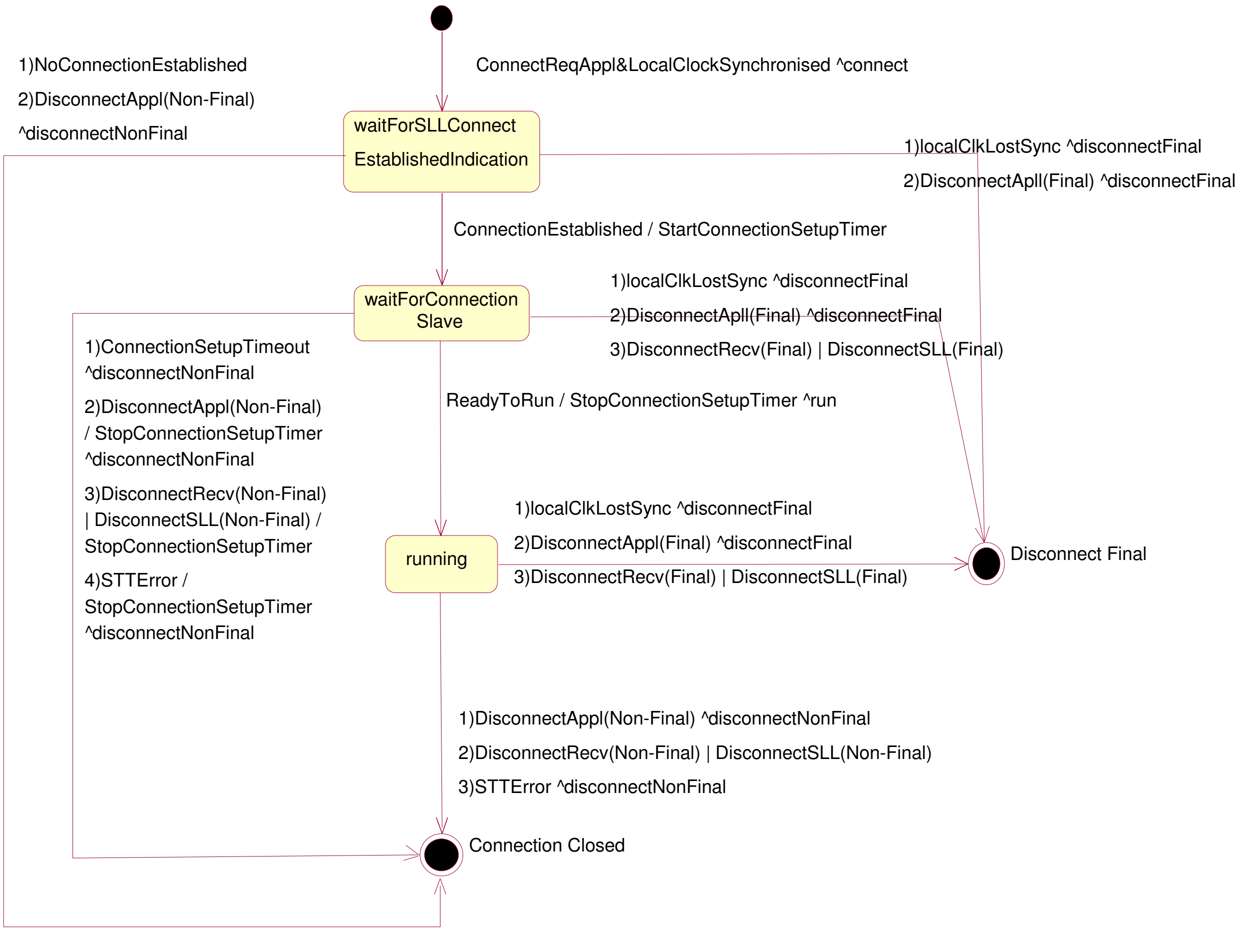
ConnectReqAppl&LocalClockSynchronised ^connect
1)NoConnectionEstablished
2)DisconnectAppl(Non-Final)
^disconnectNonFinal
waitForSLLConnect EstablishedIndication
1)localClkLostSync ^disconnectFinal
2)DisconnectApll(Final) ^disconnectFinal
ConnectionEstablished / StartConnectionSetupTimer
waitForConnection Slave
1)localClkLostSync ^disconnectFinal
2)DisconnectApll(Final) ^disconnectFinal
3)DisconnectRecv(Final) | DisconnectSLL(Final)
1)ConnectionSetupTimeout ^disconnectNonFinal
2)DisconnectAppl(Non-Final) / StopConnectionSetupTimer ^disconnectNonFinal
3)DisconnectRecv(Non-Final) | DisconnectSLL(Non-Final) / StopConnectionSetupTimer
4)STTError / StopConnectionSetupTimer ^disconnectNonFinal
ReadyToRun / StopConnectionSetupTimer ^run
running
1)localClkLostSync ^disconnectFinal
2)DisconnectAppl(Final) ^disconnectFinal
3)DisconnectRecv(Final) | DisconnectSLL(Final)
Disconnect Final
1)DisconnectAppl(Non-Final) ^disconnectNonFinal
2)DisconnectRecv(Non-Final) | DisconnectSLL(Non-Final)
3)STTError ^disconnectNonFinal
Connection Closed

Figure 11 Logical Connection Master state machine

7.6.1.3.1 States shall be:

7.6.1.3.1.1 waitForSLLConnectEstablishedIndication(state): Waiting for the logical connection to be established by the SLL.

7.6.1.3.1.2 WaitForConnectionSlave (state): Wait for Ready to Run message from Logical Connection Slave.

7.6.1.3.1.3 Running (state): connection opened.

7.6.1.3.1.4 Connection closed (state):The connection is closed, the connection instance is cancelled, but may be setup again.

7.6.1.3.1.5 Disconnect Final (state): connection finally closed, no re-connection possible.

7.6.1.3.2 Events shall be:

7.6.1.3.2.1 ConnectReqAppl&LocalClockSynchronised (event) : Local Clock is synchronised and Request from application to connect was received.

7.6.1.3.2.2 ConnectionSetupTimeout (event): Timeout of ConnectionSetupTimer.

7.6.1.3.2.3 ReadyToRun (event): reception of Ready To Run message from Logical Connection Slave.

7.6.1.3.2.4 DisconnectRecv(local disconnect_type) (event): reception of a Disconnect Telegram from Logical Connection Slave. In case the disconnect_type of the received telegram is Final, the local disconnect_type may be Final or Non-Final, depending on implementation. Otherwise it is Non-Final.

7.6.1.3.2.5 DisconnectAppl(disconnect_type) (event): disconnect of given disconnect_type is requested by application.

7.6.1.3.2.6 DisconnectSLL(disconnect_type) (event): disconnect of given disconnect_type by the SLL.

7.6.1.3.2.7 ConnectionEstablished (event): Logical connection established by the SLL.

7.6.1.3.2.8 localClkLostSync (event): Local Clock in the node get unsynchronised.

7.6.1.3.2.9 NoConnectionEstablished (event): Reception of an indication from the SLL, indicating that the logical connection could not be opened.

7.6.1.3.2.10 STTError (event): Safety Time Tolerance for telegram received from Logical Connection Slave exceeded.

7.6.1.3.3 Actions shall be:

7.6.1.3.3.1 StartConnectionSetupTimer (action): set ConnectionSetupTimer to ConnectionSetupTimeLimit.

7.6.1.3.3.2 StopConnectionSetupTimer (action): Stop ConnectionSetupTimer.

7.6.1.3.4 Send-clauses shall be:

7.6.1.3.4.1 Connect (send-clause): send a Connect Request Telegram to the Logical Connection Slave.

7.6.1.3.4.2 run (send-clause): send a Run Telegram to the Logical Connection Slave.

7.6.1.3.4.3 disconnectFinal (send-clause): send a (Final) Disconnect Telegram to the Logical Connection Slave.

7.6.1.3.4.4 disconnectNonFinal (send-clause): send a (Non-Final) Disconnect Telegram to the Logical Connection Slave.

7.7 Logical Connection Slave

7.7.1.1 This chapter specifies the requested behaviour of a Logical Connection Slave at the interface between the connected nodes.

7.7.1.1.1 Note: Every Logical connection needs a separate state machine.

7.7.1.2 If the Logical Connection Slave receives an indication that the connection has been established by the Safe Link Layer, and the Local Clock is not synchronised then:

7.7.1.2.1 a Non-Final disconnect Telegram shall be sent in case the local clock is not yet synchronised and

7.7.1.2.2 no telegram shall be sent in case the synchronisation is lost and the connection shall not be opened because the node is isolated.

7.7.1.3

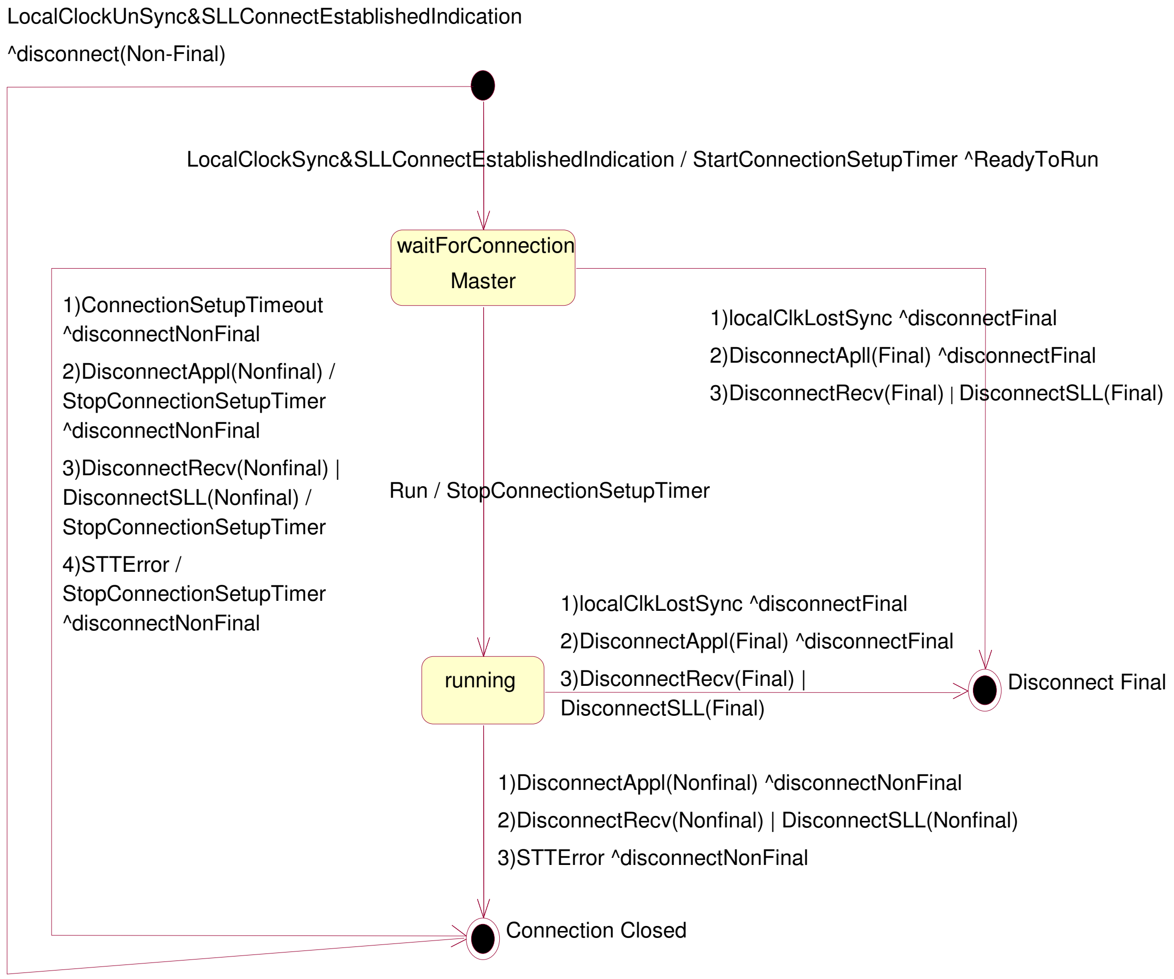
LocalClockUnSync&SLLConnectEstablishedIndication
^disconnect(Non-Final)
LocalClockSync&SLLConnectEstablishedIndication / StartConnectionSetupTimer ^ReadyToRun
waitForConnection Master
1)ConnectionSetupTimeout ^disconnectNonFinal
2)DisconnectAppl(Nonfinal) / StopConnectionSetupTimer ^disconnectNonFinal
3)DisconnectRecv(Nonfinal) | DisconnectSLL(Nonfinal) / StopConnectionSetupTimer
4)STTError / StopConnectionSetupTimer ^disconnectNonFinal
1)localClkLostSync ^disconnectFinal
2)DisconnectApll(Final) ^disconnectFinal
3)DisconnectRecv(Final) | DisconnectSLL(Final)
Run / StopConnectionSetupTimer
1)localClkLostSync ^disconnectFinal
2)DisconnectAppl(Final) ^disconnectFinal
3)DisconnectRecv(Final) | DisconnectSLL(Final)
running
Disconnect Final
1)DisconnectAppl(Nonfinal) ^disconnectNonFinal
2)DisconnectRecv(Nonfinal) | DisconnectSLL(Nonfinal)
3)STTError ^disconnectNonFinal
Connection Closed

Figure 12 Logical Connection Slave state machine

7.7.1.3.1 States shall be:

7.7.1.3.1.1 WaitForConnectionMaster (state): Wait for Run message from the Logical Connection Master.

7.7.1.3.1.2 Running (state): Connection opened.

7.7.1.3.1.3 Connection closed (state): The connection is closed, the connection instance is cancelled, but may be setup again.

7.7.1.3.1.4 Disconnect Final (state): connection finally closed, no re-connection possible.

7.7.1.3.2 Events shall be:

7.7.1.3.2.1 LocalClockSync&SLLConnectEstablishedIndication (event): Local Clock is synchronised, and logical connection established by SLL.

7.7.1.3.2.2 ConnectionSetupTimeout (event): Timeout of ConnectionSetupTimer.

7.7.1.3.2.3 run (event): Receive a Run Message from the Logical Connection Master.

7.7.1.3.2.4 LocalClkLostSync (event): Local Clock in the node gets unsynchronised.

7.7.1.3.2.5 DisconnectRecv(local disconnect_type) (event): reception of a Disconnect Telegram from the Logical Connection Master.

7.7.1.3.2.6 In case the disconnect_type of the received telegram is Final, the local disconnect_type may be Final or Non-Final, depending on implementation. Otherwise it is Non-Final.

7.7.1.3.2.7 DisconnectAppl(disconnect_type) (event): disconnect of given disconnect_type is requested by application.

7.7.1.3.2.8 DisconnectSLL(disconnect_type) (event): disconnect of given disconnect_type by the SLL.

7.7.1.3.2.9 STTError (event): Safety Time Tolerance for telegram received from Logical Connection Master exceeded.

7.7.1.3.2.10 LocalClockUnSync&SLLConnectEstablishedIndication (event): Local Clock is not yet synchronised, and logical connection established by SLL.

7.7.1.3.3 Actions shall be:

7.7.1.3.3.1 StartConnectionSetupTimer (action): set a timer to ConnectionSetupTimeLimit.

7.7.1.3.3.2 StopConnectionSetupTimer (action): Stop ConnectionSetupTimer.

7.7.1.3.4 Send-clauses shall be:

7.7.1.3.4.1 ReadyToRun (send-clause): send a Ready to Run Message to the Logical Connection Master.

7.7.1.3.4.2 disconnectFinal (send-clause): send a (Final) Disconnect Telegram to the Logical Connection Master.

7.7.1.3.4.3 disconnectNonFinal (send-clause): send a (Non-Final) Disconnect Telegram to the Logical Connection Master.

7.8 Time Validation of Received Messages

7.8.1.1 This procedure shall be applied both to reception of point-to-point messages and multicast messages.

7.8.1.2 The following messages shall be validated:

7.8.1.2.1 Ready To Run message (point-to-point)

7.8.1.2.2 Run message (point-to-point)

7.8.1.2.3 Application messages (point-to-point and multicast)

7.8.1.3 This procedure shall not apply during clock initialisation.

7.8.1.4 Intentionally deleted

7.8.1.5 In order to monitor and control the age of a received message the received STL_Time_Stamp shall be validated to the conditions in 7.8.1.14.

7.8.1.6 The value of STTmin and STTmax shall be calculated for each logical connection according to the formulas given in 7.8.1.7 – 7.8.1.11 below.

7.8.1.6.1 Justification: Senders may have different performance, or the application data has different requirements.

7.8.1.7 STTmin = Static_Transfer_Time – LCI

7.8.1.8 STTmax = Static_Transfer_Time + Dynamic_Transfer_Time + LCI

7.8.1.9 Note: LCI is regarded in the respective receiving node.

7.8.1.10 Static_Transfer_Time = Sender_Static_Transfer_Time +
Receiver_Static_Transfer_Time + Static_Bus_Transfer_Time

7.8.1.11 Dynamic_Transfer_Time = Sender_Dynamic_Transfer_Time +
Receiver_Dynamic_Transfer_Time + Dynamic_Bus_Transfer_Time

7.8.1.12 The transfer time delays can be positive or negative. Calculation shall be able to handle positive and negative result.

7.8.1.13 Note: The Sender_Static_Transfer_Time and Sender_Dynamic_Transfer_Time are sent to the receiver at start-up of the Safe Time Layer.

7.8.1.14 Conditions are:

7.8.1.14.1 Minimum time age: Local_Time_Stamp - STL_Time_Stamp > STTmin

7.8.1.14.2 Maximum time age: Local_Time_Stamp - STL_Time_Stamp < STTmax

7.8.1.14.3 Local_Time_Stamp in 7.8.1.14.1 and 7.8.1.14.2 shall be the value of Local_Reference_Time at reception of a telegram.

7.8.1.14.4 If the conditions are not met, there is a time error, so the message shall be rejected. For a point-to-point connection a Non-Final disconnection shall be performed for this connection.

8. INTENTIONALLY DELETED

9. LIST OF CONSTANTS

9.1.1.1 This chapter gives a list of the constants used within the Safe Time Layer together with their definition. The numeric values are specified in SUBSET-059 (Performance Requirements for STMs).

9.2 LocalClockMaxReSyncInterval

9.2.1.1 Maximum time between two successfully executed resynchronisations (see chapter 7.4.1.6.1).

9.3 SafeTimeLayerStartupInterval

9.3.1.1 Initial interval between STL Startup messages until StartupSynchronisationTimeLimit has expired (see chapter 6.6.1.3).

9.4 SafeTimeLayerReStartInterval

9.4.1.1 Later interval between STL Startup messages after StartupSynchronisationTimeLimit has expired (see chapter 6.6.1.4).

9.5 StartupSynchronisationTimeLimit

9.5.1.1 Time when the sending interval of the Safe Time Layer Startup message changes from SafeTimeLayerStartupInterval to SafeTimeLayerReStartInterval (see chapter 6.6.1.4).

9.6 MaxClockInaccuracyAfterAdjustFactor

9.6.1.1 This constant gives the maximum allowed error of Local_Reference_Time in relation to RefTime. See 7.4.3.5.

9.6.1.2 Note: It does not define the maximum inaccuracy of the Local_Reference_Time compared to the Reference Time on the bus. It limits the short term drift.

9.7 TimeForLongTermDriftCheck

9.7.1.1 Non floating time interval for checking the long term drift (see chapters 7.4.3.6 and 7.4.3.7).

9.8 MinNumberofSyncAndRefMsgReceived

9.8.1.1 Minimum number of valid received Sync and Reference Time messages during a long term drift check interval (see chapter 7.4.3.6).

9.9 SyncAndRefTimeSyncInterval

9.9.1.1 Interval between sending Sync and Reference Time messages at start up for clock synchronisation (see chapter 7.3.1.4).

9.10 SyncAndRefTimeRunInterval

9.10.1.1 Interval between sending Sync and Reference Time messages after start up (see chapter 7.3.1.5).

9.11 SyncAndRefTimeStartupTimeLimit

9.11.1.1 Time when the sending interval of the Sync and Reference Time message changes from SyncAndRefTimeSyncInterval to SyncAndRefTimeRunInterval (see chapter 7.3.1.5).

9.12 ConnectionSetupTimeLimit

9.12.1.1 Time limit for setup (establishment) of a STL connection (see chapters 7.6.1.3.3.1 and 7.7.1.3.3.1).

10. APPENDIX: SERVICES PROVIDED BY THE SAFE TIME LAYER

10.1.1.1 This appendix is informal and shall be read as an example.

10.1.1.2 The Safe Time Layer may provide the following services for the application:

10.1.1.2.1

Service	**Description**	**Parameters**	**Return**
Connect	Initialises a logical connection	The parameters required by the connect service in the Safe Link Layer and; Sender_Dynamic_Transfer_Time or Receiver_Dynamic_Transfer_Time; Sender_Static_Transfer_Time or Receiver_Static_Transfer_Time	Logical connection ID or Fail
Get Data	Returns received application data	Logical connection ID	Application data, No data or Fail; Reception Time Stamp
Send Data	Send application data	Logical connection ID; Application data	OK or Fail; Send time stamp

Disconnect	Disconnect a logical connection	Logical connection ID;
		Reason for disconnect;
		Type of disconnect;
		Reason for disconnect text
			OK or Fail

Logical connection active	Ask if a logical connection is active	Logical connection ID	Connected, Connecting, Connection closed, Final Disconnected or Fail;
			Reason for disconnect;
			Reason for disconnect text

Get time	Returns the local clocks estimate of the reference time.	None	Current Reference Time, Unsynchronised or Fail

11. APPENDIX: DEFINITION OF NOTATION

11.1 State Machine

11.1.1.1 In this document the following notation for state machine view is used:

11.1.1.2

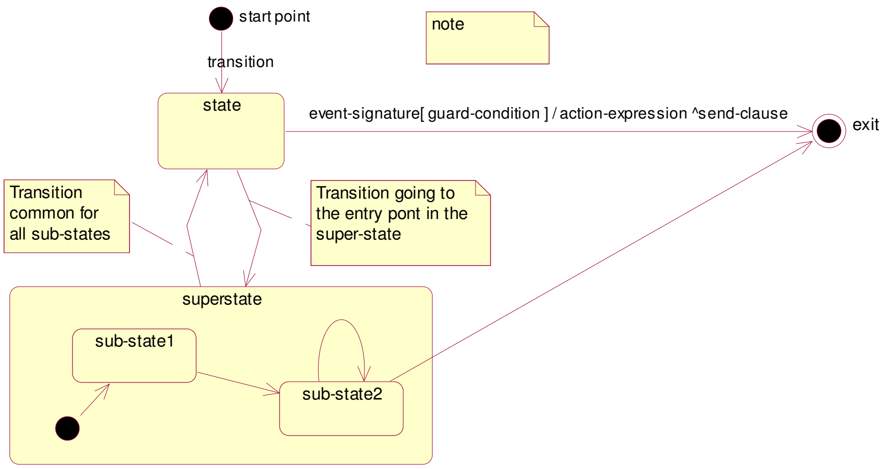
start point
note
transition
state
event-signature[guard-condition] / action-expression ^send-clause
exit
Transition common for all sub-states
Transition going to the entry pont in the super-state
superstate
sub-state1
sub-state2

Figure 13 State machine notation

11.1.1.2.1 event-signature: Name of the event condition that shall be fulfilled to trig the transition.

11.1.1.2.2 [guard-condition]: The guard-condition must be true to enable evaluation of the event condition named by the event-signature.

11.1.1.2.3 /action-expression: The action that is evaluated at the transition.

11.1.1.2.4 ^send-clause: The name of the message sent at the transition.

11.2 Sequence Chart

11.2.1.1 In this document the following notation for the Sequence Chart view is used:

11.2.1.2

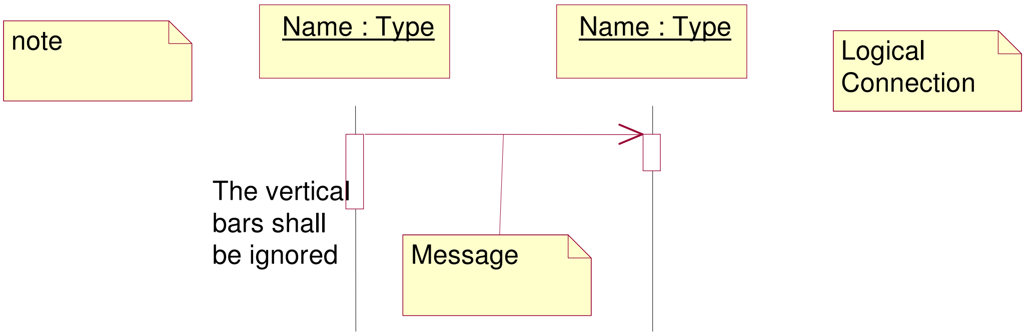

Figure 14 Sequence Chart notation

12. INTENTIONALLY DELETED