60802 Time Sync – Clocks Relationships & Normative Requirements

David McCall – Intel Corporation

Version 1

Shared last time…

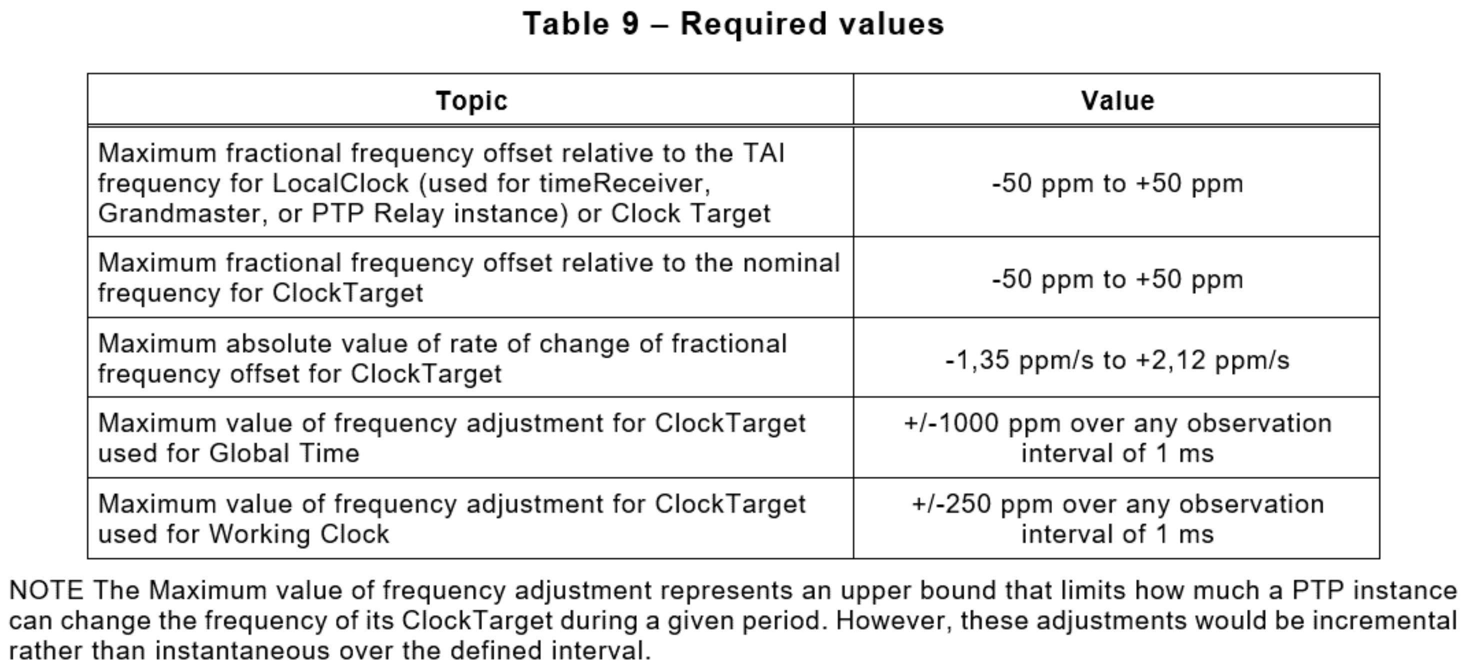


Table 9 – Required values

Topic	Value
Maximum fractional frequency offset relative to the TAI frequency for LocalClock (used for timeReceiver, Grandmaster, or PTP Relay instance) or Clock Target	-50 ppm to +50 ppm
Maximum fractional frequency offset relative to the nominal frequency for ClockTarget	-50 ppm to +50 ppm
Maximum absolute value of rate of change of fractional frequency offset for ClockTarget	-1,35 ppm/s to +2,12 ppm/s
Maximum value of frequency adjustment for ClockTarget used for Global Time	+/-1000 ppm over any observation interval of 1 ms
Maximum value of frequency adjustment for ClockTarget used for Working Clock	+/-250 ppm over any observation interval of 1 ms

NOTE The Maximum value of frequency adjustment represents an upper bound that limits how much a PTP instance can change the frequency of its ClockTarget during a given period. However, these adjustments would be incremental rather than instantaneous over the defined interval.

ClockSource/Target & timeTransmitter/Receiver APIs

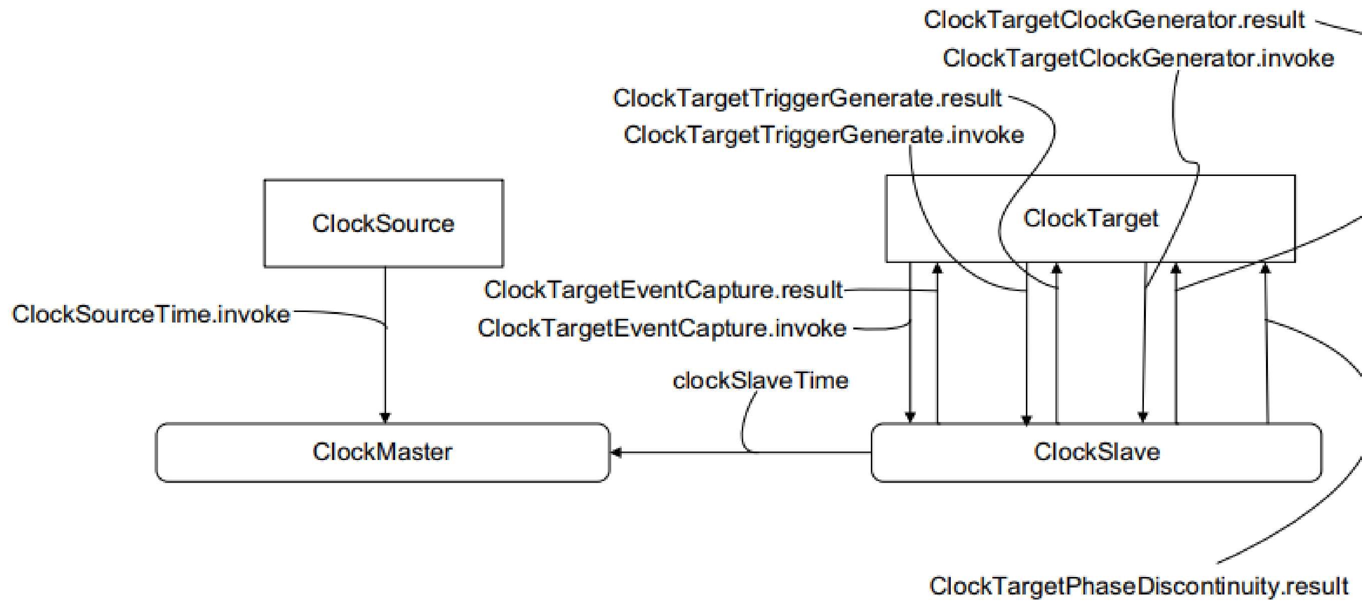
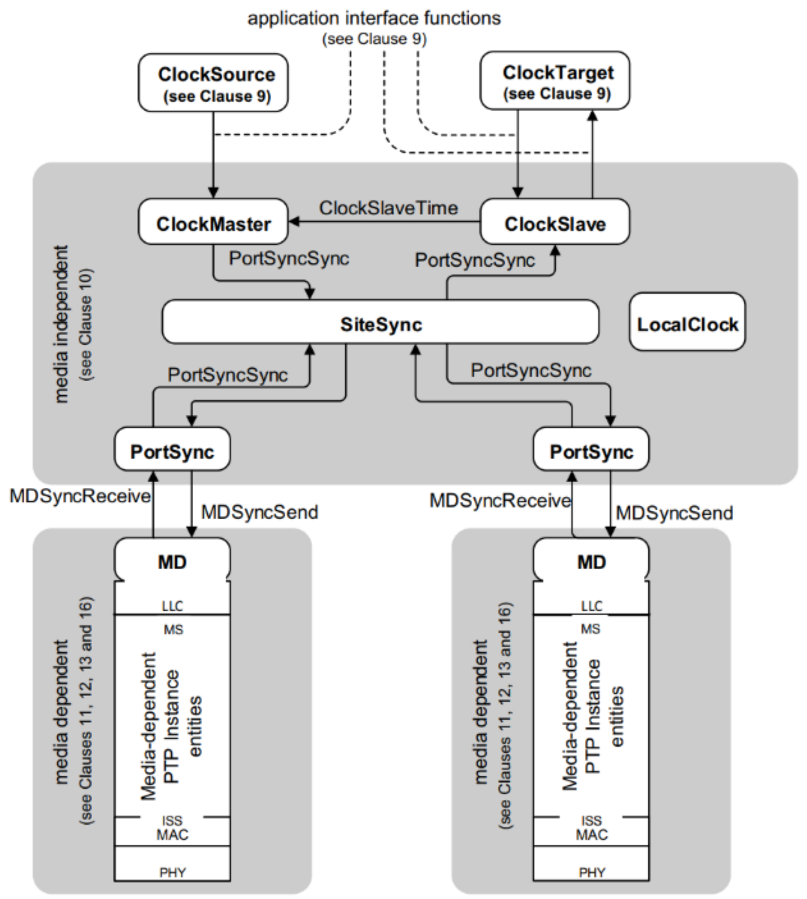

ClockTargetClockGenerator.result
ClockTargetClockGenerator.invoke
ClockTargetTriggerGenerate.result
ClockTargetTriggerGenerate.invoke
ClockSource
ClockTarget
ClockTargetEventCapture.result
ClockSourceTime.invoke
ClockTargetEventCapture.invoke
clockSlaveTime
ClockMaster
ClockSlave
ClockTargetPhaseDiscontinuity.result

Figure 9-1—Application interfaces

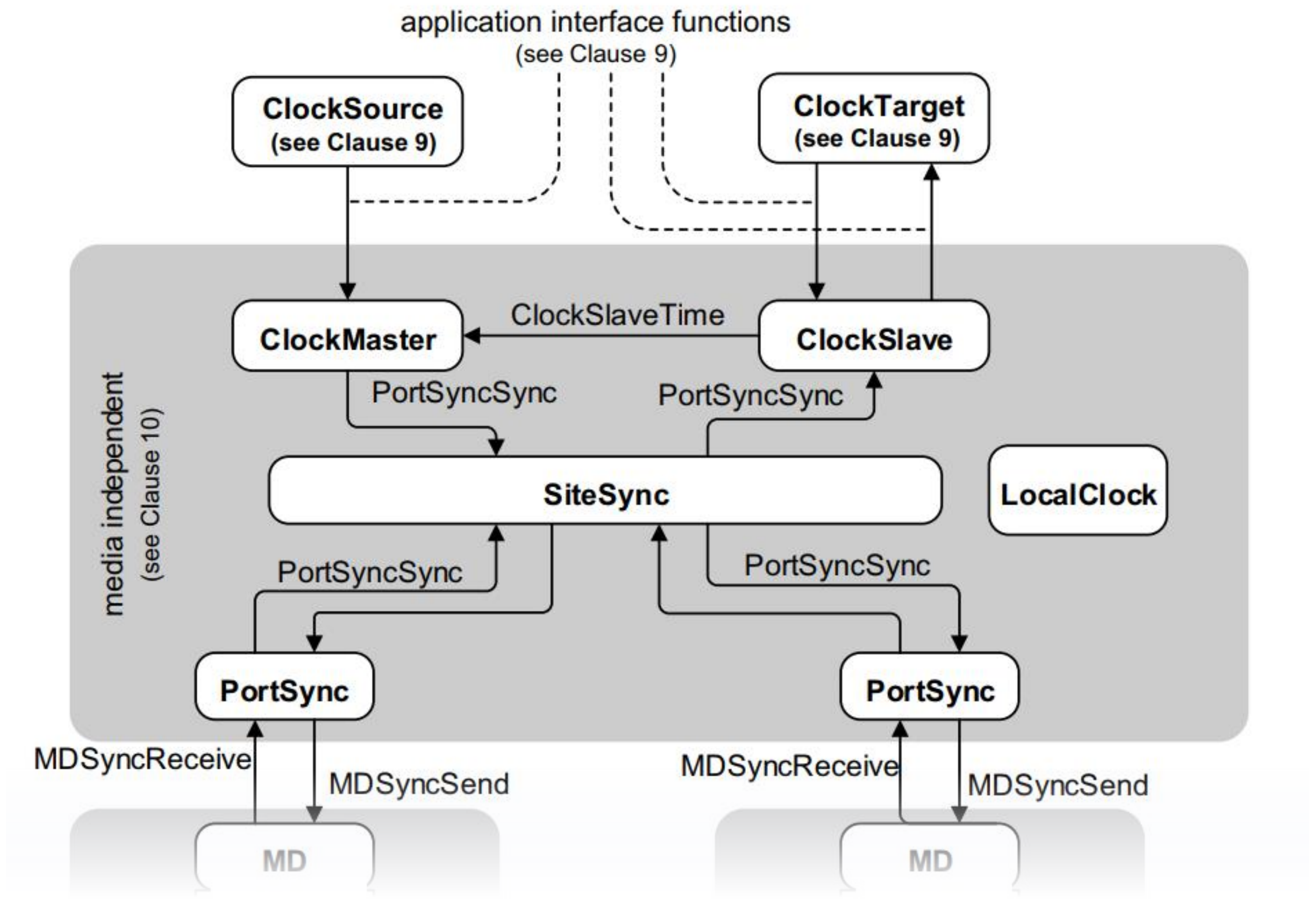
timeTransmitter/Receiver & Local Clock



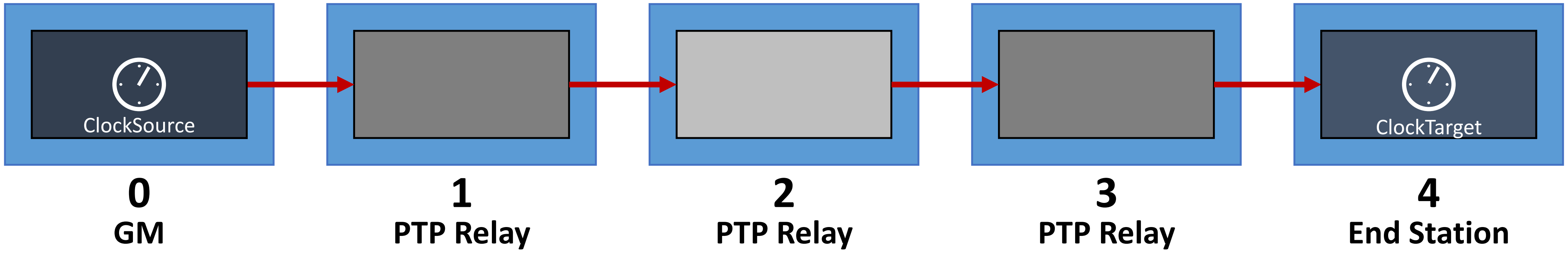
application interface functions
(see Clause 9)
ClockSource (see Clause 9)
ClockTarget (see Clause 9)
ClockMaster
ClockSlaveTime
ClockSlave
PortSyncSync
PortSyncSync
SiteSync
LocalClock
media independent (see Clause 10)
PortSyncSync
PortSyncSync
PortSync
PortSync
MDSyncReceive
MDSyncSend
MDSyncReceive
MDSyncSend
MD
LLC
MS
Media-dependent PTP Instance entities
ISS
MAC
PHY
media dependent (see Clauses 11, 12, 13 and 16)
MD
LLC
MS
Media-dependent PTP Instance entities
ISS
MAC
PHY
media dependent (see Clauses 11, 12, 13 and 16)

Figure 10-1—Model for media-independent layer of PTP Instance

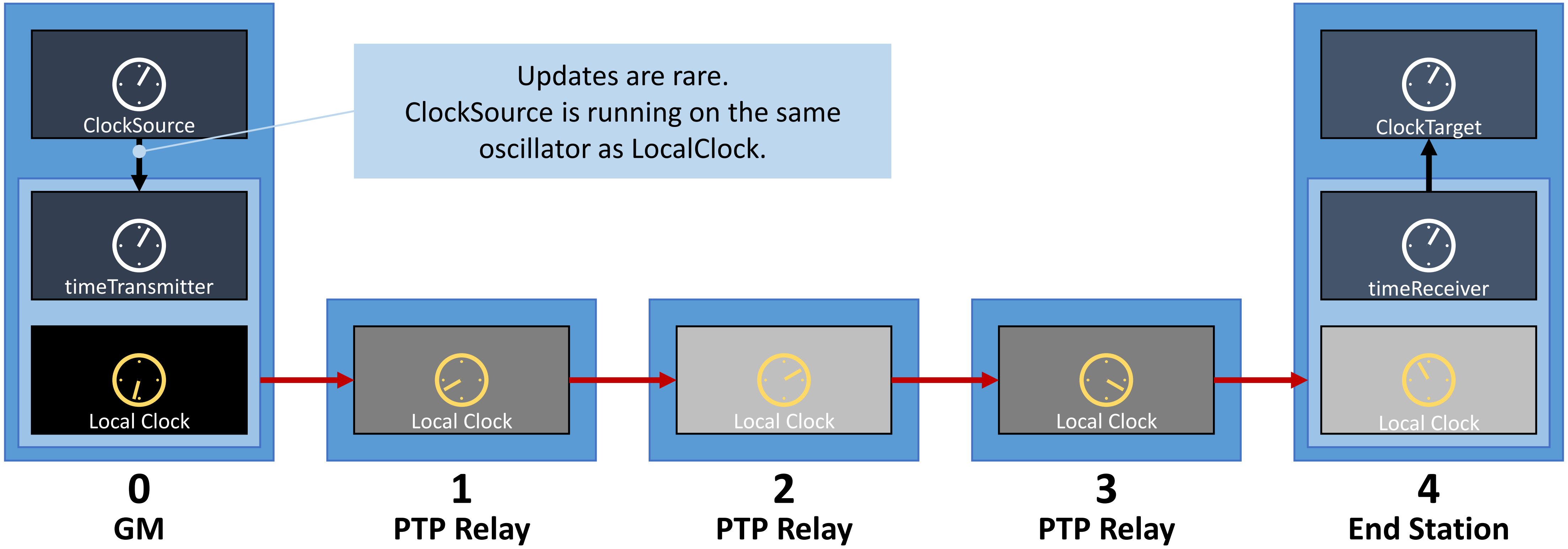
timeTransmitter/Receiver & Local Clock



Independent GM

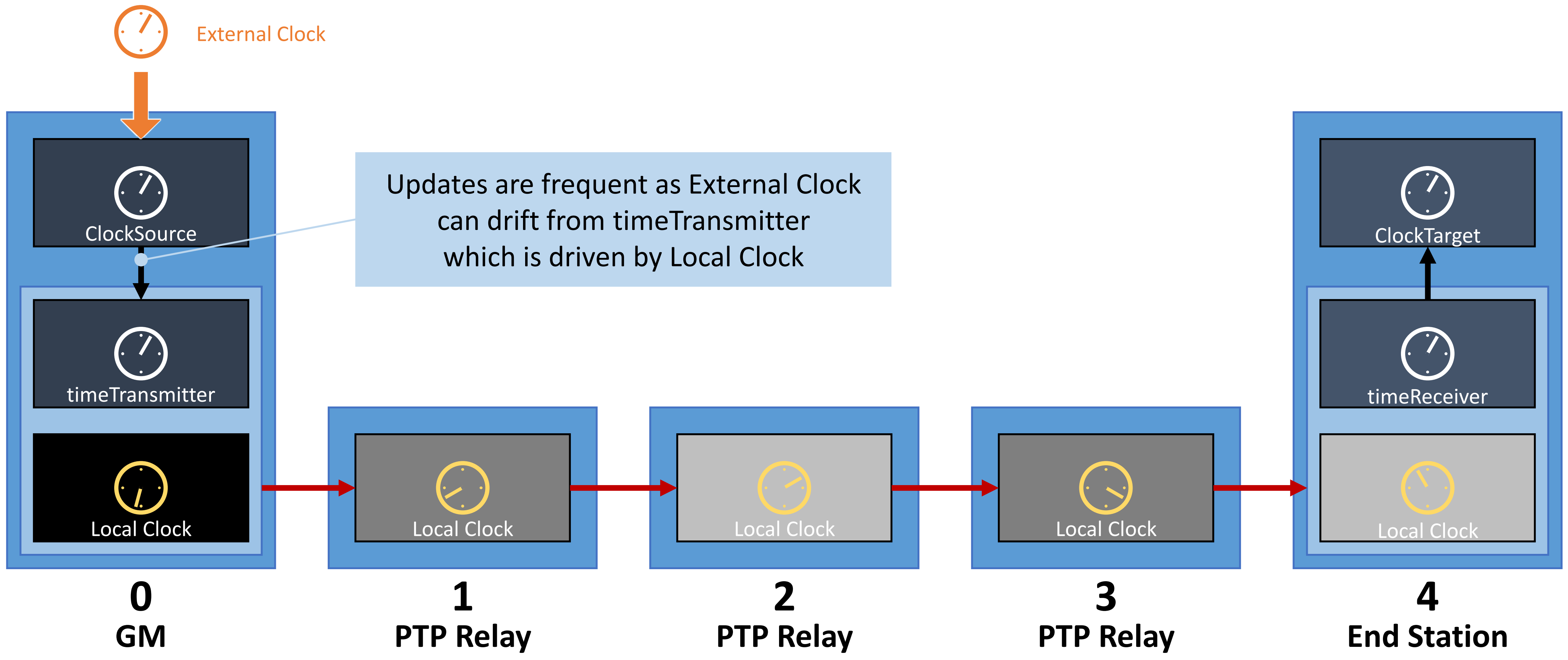


Independent GM



Updates are rare.
ClockSource is running on the same oscillator as LocalClock.

Externally Controlled GM

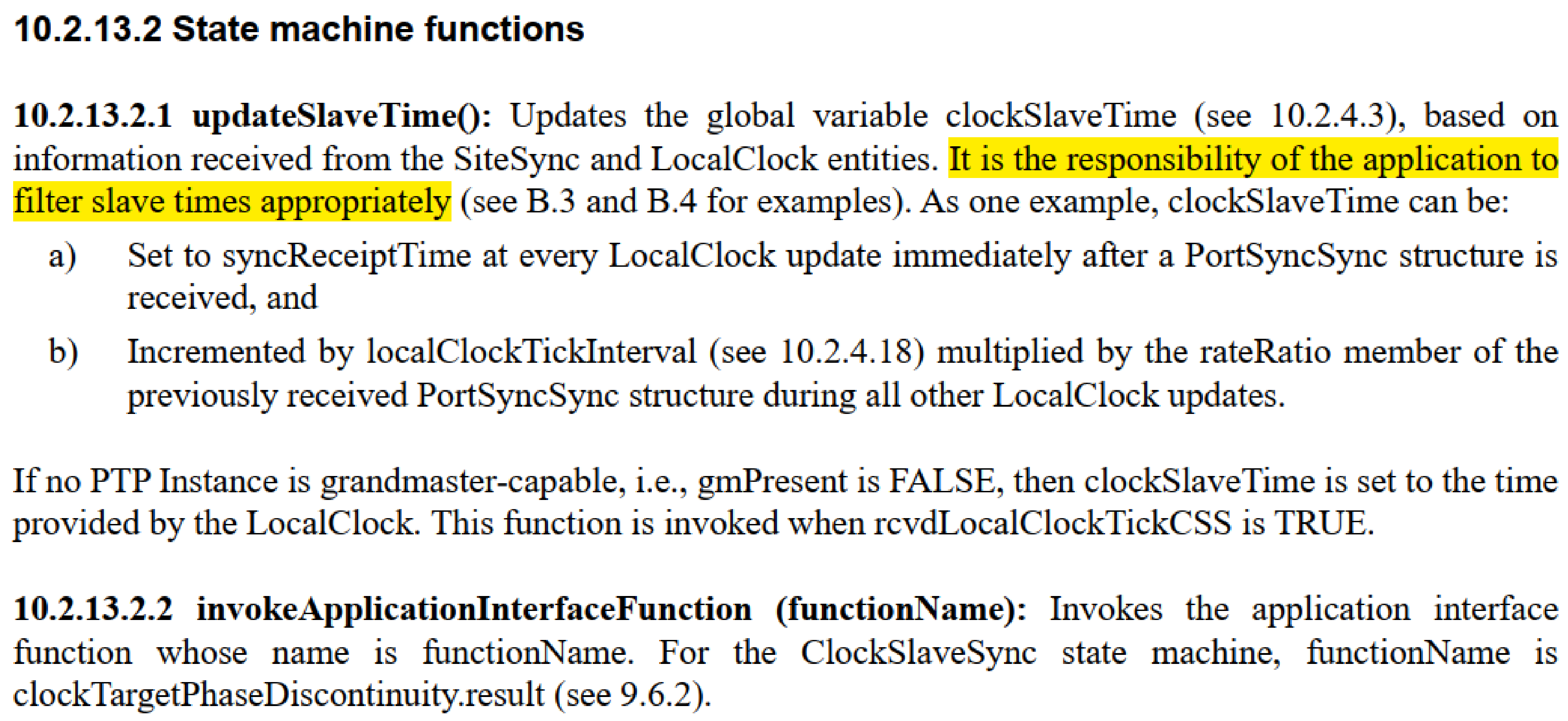


Updates are frequent as External Clock can drift from timeTransmitter which is driven by Local Clock

For discussion (and written to provoke it!)…

- 802.1AS defines timeTransmitter & timeReceiver as state machines, not as oscillators.
  - Every node has both state machines. In a PTP Relay and End Station the timeTransmitter is driven only by the timeReceiver (via clockReceiverTime)
- 802.1AS defines ClockSource & ClockTarget as entities that interface with timeTransmitter & via defined APIs, not as oscillators.
- Local Clock needs to be have ppm and ppm/s limits.
- It doesn’t make sense to define ppm and ppm/s limits for timeTransmitter, timeReceiver, ClockSource or ClockTarget.
- The filtering / control loop requirements are normative and applied to the output of the timeTransmitter received, via the defined APIs, by the ClockTarget.
  - The error generation limits at an End Station are on the filtered values / output of the control loop. (Which is not the case for a PTP Relay instance, which operates based on the Local Clock only. For that, the normative requirements on error generation apply to the Correction Field & Rate Ratio output.)

802.1AS-2020 10.2.13 Clock~~Slave~~ReceiverSync state machine



10.2.13.2 State machine functions

10.2.13.2.1 updateSlaveTime(): Updates the global variable clockSlaveTime (see 10.2.4.3), based on information received from the SiteSync and LocalClock entities. It is the responsibility of the application to filter slave times appropriately (see B.3 and B.4 for examples). As one example, clockSlaveTime can be:

a) Set to syncReceiptTime at every LocalClock update immediately after a PortSyncSync structure is received, and

b) Incremented by localClockTickInterval (see 10.2.4.18) multiplied by the rateRatio member of the previously received PortSyncSync structure during all other LocalClock updates.

If no PTP Instance is grandmaster-capable, i.e., gmPresent is FALSE, then clockSlaveTime is set to the time provided by the LocalClock. This function is invoked when rcvdLocalClockTickCSS is TRUE.

10.2.13.2.2 invokeApplicationInterfaceFunction (functionName): Invokes the application interface function whose name is functionName. For the ClockSlaveSync state machine, functionName is clockTargetPhaseDiscontinuity.result (see 9.6.2).

For discussion (and written to provoke it!)…

- When ClockSource is driven by an External Clock, drift between External Clock and GM Local Clock **may** or **may not** cause additional errors at the ClockTarget
  - Depends on how frequently the timeTransmitter is updated via the ClockSourceTime.invoke API is called.
  - Frequent updates relative to Sync Interval → minimal additional errors
  - Infrequent updates relative to Sync Interval → additional errors based on difference between External Clock offset + GM Local Clock offset
    - If GM Local Clock is running faster than External Clock (e.g. + 10 ppm), even if stable (i.e. 0 ppm/s) an error will build up until the next time timeTransmitter is updated
    - Note that **ppm/s isn't the critical parameter** regarding the **additional** error, as the External Clock isn't part of the PTP process, e.g. there is no mechanism to measure the rate ratio between the GM Local Clock and the External Clock.
    - However, ppm/s is still a source of error. If the External Clock is drifting, it will have the same impact as the GM Local Clock drifting.
- Should we specify a normative requirement on how frequently the ClockSource should be updated to minimise the additional source of error?
- To ensure the same time sync accuracy (±1 µs at node 100) as an independent system, an External Clock should have the same ppm/s limits as the GM Local Clock, but…
- The ClockTarget at node 100 of a 60802 system will have slightly worse ppm/s characteristics than the GM Local Clock, so…should we calculate (via simulation) how much worse…and how much buffer there is…and therefore how many systems can be chained together while still maintaining accuracy relative to the first system's ClockSource / timeTransmitter?

Thank you!

Time Sync – Elements & Relationships

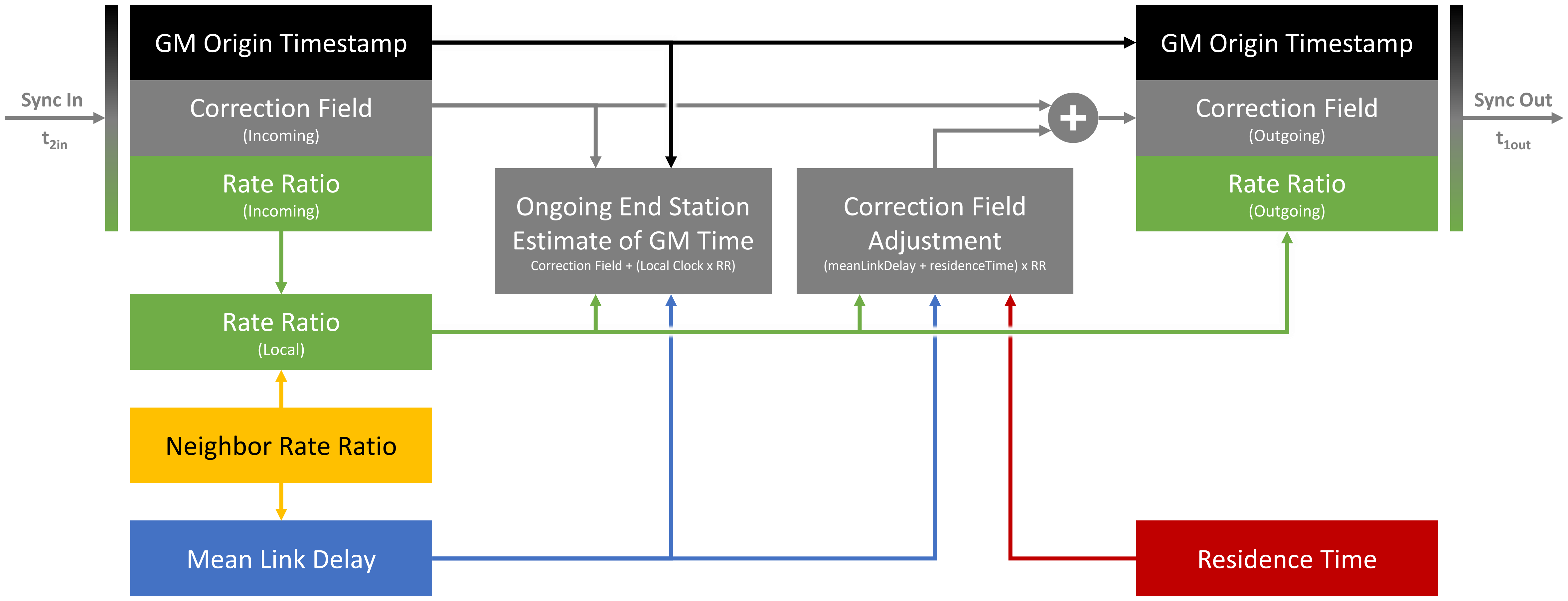
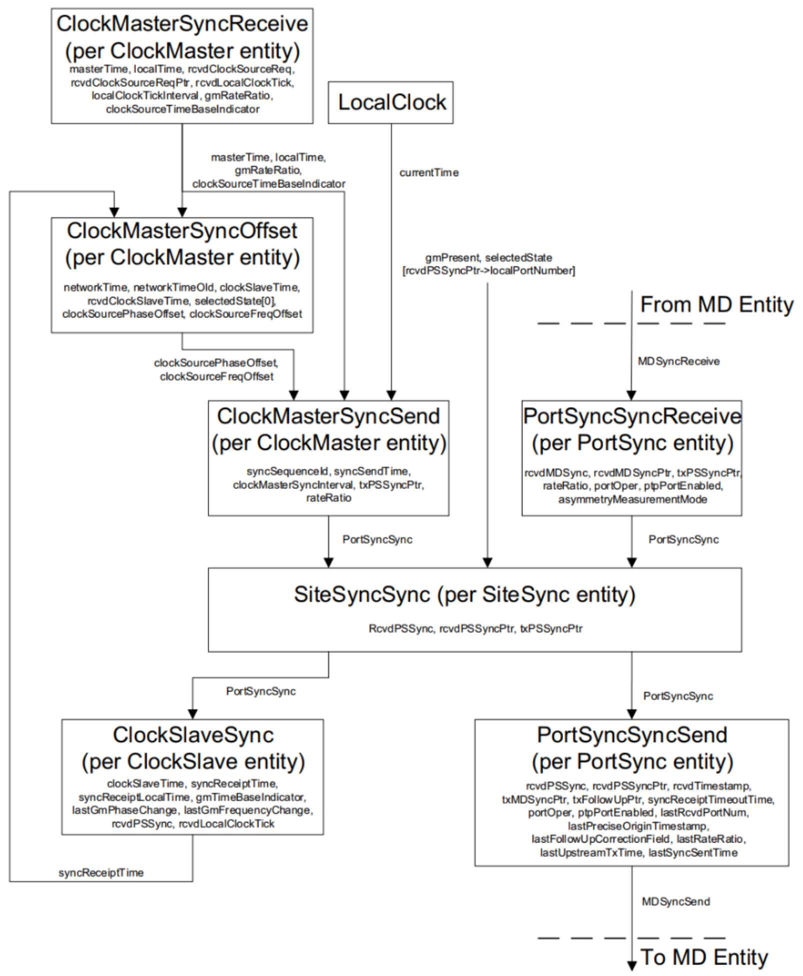


Figure 10-2—Time-synchronization state machines—overview and interrelationships


ClockMasterSyncReceive (per ClockMaster entity)
masterTime, localTime, rcvdClockSourceReq, rcvdClockSourceReqPtr, rcvdLocalClockTick, localClockTickInterval, gmRateRatio, clockSourceTimeBaseIndicator
LocalClock
masterTime, localTime, gmRateRatio, clockSourceTimeBaseIndicator
currentTime
ClockMasterSyncOffset (per ClockMaster entity)
networkTime, networkTimeOld, clockSlaveTime, rcvdClockSlaveTime, selectedState[0], clockSourcePhaseOffset, clockSourceFreqOffset
gmPresent, selectedState [rcvdPSSyncPtr->localPortNumber]
From MD Entity
MDSyncReceive
clockSourcePhaseOffset, clockSourceFreqOffset
ClockMasterSyncSend (per ClockMaster entity)
syncSequenceId, syncSendTime, clockMasterSyncInterval, txPSSyncPtr, rateRatio
PortSyncSyncReceive (per PortSync entity)
rcvdMDSync, rcvdMDSyncPtr, txPSSyncPtr, rateRatio, portOper, ptpPortEnabled, asymmetryMeasurementMode
PortSyncSync
PortSyncSync
SiteSyncSync (per SiteSync entity)
RcvdPSSync, rcvdPSSyncPtr, txPSSyncPtr
PortSyncSync
PortSyncSync
ClockSlaveSync (per ClockSlave entity)
clockSlaveTime, syncReceiptTime, syncReceiptLocalTime, gmTimeBaseIndicator, lastGmPhaseChange, lastGmFrequencyChange, rcvdPSSync, rcvdLocalClockTick
PortSyncSyncSend (per PortSync entity)
rcvdPSSync, rcvdPSSyncPtr, rcvdTimestamp, txMDSyncPtr, txFollowUpPtr, syncReceiptTimeoutTime, portOper, ptpPortEnabled, lastRcvdPortNum, lastPreciseOriginTimestamp, lastFollowUpCorrectionField, lastRateRatio, lastUpstreamTxTime, lastSyncSentTime
syncReceiptTime
MDSyncSend
To MD Entity