PTP Clock Manager for Linux Management API

This document lists the functions and type definitions of the C application programming interface to access and configure the PTP Clock Manager (pcm4l) software.

Contents

1. Introduction

The file `idtCore/management/include/mngApi/mngApi.h` contains the relevant header definitions to be included by the calling software for the `pcm4l` management API functions.

2. Type Definitions

The following type definitions are listed in alphabetical order.

2.1 T_CallbackFunctionPointer

idtCommon/include/messageLog/cmnCallbackLog.h

```
typedef void (*T_CallbackFunctionPointer)( T_cmnMessageData const * );
```

2.2 T_cmnErrorCode

idtCommon/include/cmnErrorCode.h

```
typedef enum
{
  E_cmnErrorCode_OK                     = 0,  /* Command was successful */
  E_cmnErrorCode_ResponseTimeout        = 1,  /* Response message was not received before
                                                 response timeout */
  E_cmnErrorCode_FunctionNotSupported   = 2,  /* Function not supported */
  E_cmnErrorCode_InvalidMaster          = 3,  /* Invalid master */
  E_cmnErrorCode_NoTextStringFound      = 4,  /* No corresponding text string mapped */
  E_cmnErrorCode_NotConfigured          = 5,  /* Value was not configured */
  E_cmnErrorCode_NotAccepted            = 6,  /* Action not accepted, function busy */
  E_cmnErrorCode_InvalidValue           = 7,  /* Value is invalid */
  E_cmnErrorCode_CallbackNotRegistered  = 8,  /* Callback function is not registered */
  E_cmnErrorCode_TimerFailed            = 9,  /* Timer failed */
  E_cmnErrorCode_CallbackReturnsFailure = 10, /* Callback function returns failure */
  E_cmnErrorCode_BestMasterNotFound     = 11, /* <Best master not found */
  E_cmnErrorCode_FailedToRetrievePortID = 12, /* Failed to retrieve the port ID (clockID) */
  E_cmnErrorCode_InvalidApiCommand      = 13, /* Invalid API Command */
  E_cmnErrorCode_GeneralError           = 14, /* General Error */
  E_cmnErrorCode_IncorrectParameters    = 15, /* Parameters are not correct */
  E_cmnErrorCode_Max
} T_cmnErrorCode;
```

2.3 T_cmnLogDescription

idtCommon/include/messageLog/cmnMessageLog.h

```
typedef struct
{
  T_cmnLogId id;
  T_cmnLogType type;
  T_cmnMessageLevelRegister messageLevelMask;
  char description[ CMN_LOGDESCRIPTION_LIMIT ];
} T_cmnLogDescription;
```

2.4 T_cmnLogId

idtCommon/include/messageLog/cmnMessageLog.h

```
typedef T_osInt16 T_cmnLogId;
```

2.5 T_cmnMessageLevelRegister

idtCommon/include/cmnTypeDef.h

```
typedef T_osUint16 T_cmnMessageLevelRegister;
```

Bit mask is the same as the JSON “selectionMask”:

```
          "selectionMask": "0000000000011111",
                                 | ||||||___ 0: Sync error
                                 | |||||____ 1: Sync warning
                                 | ||||_____ 2: Sync analysis
                                 | |||______ 3: Error
                                 | ||_______ 4: Warning
                                 | |________ 5: Debug
                                 |__________ 7: Timestamp
```

2.6 T_gnssClockCategory

idtCommon/include/cmnTypeDef.h

```
typedef enum
{
  E_GNSS_CLOCK_CATEGORY_ACTIVE = 1,
  E_GNSS_CLOCK_CATEGORY_VOID   = 2
} T_gnssClockCategory;
```

2.7 T_gnssSmState

idtCore/management/include/mngGnssSupervisor.h

```
typedef enum
{
  E_gnssState_Unqualified,
  E_gnssState_WaitFirstTodRead,
  E_gnssState_Locked,
  E_gnssState_Holdover,
  E_gnssState_Disabled,
  E_gnssState_Max
} T_gnssSmState;
```

2.8 T_ieee1588PortIdentity

```
typedef struct
{
  T_ieee1588ClockIdentity clockIdentity;
  T_osUint16 portNumber;
} T_ieee1588PortIdentity;

typedef struct
{
  T_osUint8 clockId[ IDT_CLKID_BYTE_SIZE ];
} T_ieee1588ClockIdentity;

where IDT_CLKID_BYTE_SIZE = 8
```

2.9 T_mngApiGnssStatus

idtCore/management/include/mngApi/mngApi.h

```
typedef struct
{
  T_gnssSmState  gnssStatus;
} T_mngApiGnssStatus;
```

2.10 T_mngApiLoStatus

idtCore/management/include/mngApi/mngApi.h

```
typedef struct
{
  T_mngApiLoLockStatus lockStatus;
  T_osBool qualifiedHoldover;
  T_srvLoStateId loStateId;
  T_osChar loStateName[ CMN_NAME_MAX_LENGTH ];
} T_mngApiLoStatus;

where CMN_NAME_MAX_LENGTH = 60 by default.
```

2.11 T_mngApiLoLockStatus

idtCore/management/include/mngApi/mngApi.h

```
typedef enum
{
  E_mngApiLoNeverLocked,
  E_mngApiLoFrequencyLockedOnce,
  E_mngApiLoTimeLockedOnce,
  E_mngApiLoLockStatus_Max
} T_mngApiLoLockStatus;
```

2.12 T_mngApiServoMode

idtCommon/management/include/mngApi/mngApi.h

```
typedef enum
{
  E_mngApiServoMode_Time,
  E_mngApiServoMode_Frequency,
  E_mngApiServoMode_Max
} T_mngApiServoMode;
```

2.13 T_srvLoHoldoverType

idtCommon/include/cmnTypeDef.h

```
typedef enum
{
  E_srvLoHoldoverType_Software,
  E_srvLoHoldoverType_Hardware,
  E_srvLoHoldoverType_Max
} T_srvLoHoldoverType;
```

2.14 T_srvLoStateId

idtCommon/include/cmnLoStateId.h

```
typedef enum
{
  E_srvLoInitialState = 0,
  E_srvLoUnqualifiedState = 1,
  E_srvLoLockAcqState = 2,
  E_srvLoFrequencyLockedState = 3,
  E_srvLoTimeLockedState = 4,
  E_srvLoHoldoverInSpecState = 5,
  E_srvLoHoldoverOutOfSpecState = 6,
  E_srvLoFreeRunState = 7,
  E_srvNumberLoStates = 8,
  E_srvLoStateInvalid = 9
} T_T_srvLoStateId;
```

2.15 T_srvOscillatorType

idtCommon/include/cmnOscillatorTypes.h

```
typedef enum
{
  E_srvTcxo,
  E_srvMiniOcxo,
  E_srvOcxo,
  E_srvDocxo
} T_srvOscillatorType;
```

2.16 T_srvPacketRate

```
typedef T_osDouble T_srvPacketRate;
```

2.17 T_srvPdvValues

idtCommon/include/cmnTypeDef.h

```
typedef struct
{
  T_osDouble downlink;
  T_osDouble uplink;
} T_srvPdvValues;
```

2.18 T_srvPhysicalClockCategory

```
typedef enum
{
  E_CATEGORY1        = 1,
  E_CATEGORY2        = 2,
  E_CATEGORY3        = 3,
  E_CATEGORY4        = 4,
  E_CATEGORY_DNU     = 5,     /* Do Not Use */
  E_CATEGORY_INVALID = 6
} T_srvPhysicalClockCategory;
```

2.19 T_mngApiAptsStatus

idtCore/management/include/mngApi/mngApi.h

```
typedef struct
{
  T_aptsSmState  aptsStatus;
} T_mngApiAptsStatus;
```

2.20 T_aptsSmState

idtCore/management/include/mngAptsSupervisor.h

```
typedef enum
{
  E_aptsState_GnssUnqualified,
  E_aptsState_WaitFirstTodRead,
  E_aptsState_GnssLocked,
  E_aptsState_GnssHoldover,
  E_aptsState_GnssDisabled,
  E_aptsState_Max
} T_aptsSmState;
```

3. Management API Functions

The following management API functions are grouped into categories and related functionality.

3.1 Holdover

3.1.1. mngApi_GetHoldoverLossPhysicalOosEnable

```
T_cmnErrorCode mngApi_GetHoldoverLossPhysicalOosEnable( T_osBool *holdoverLossPhysicalOosEnable );
```

DESCRIPTION

Get the holdover loss of traceability of physical layer out-of-specification enable.

ARGUMENTS

INPUTS

None

OUTPUT

holdoverLossPhysicalOosEnable – E_osTrue if enabled, E_osFalse otherwise.

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.1.2. mngApi_GetHoldoverOutOfSpecUserDefinedFrequencyOffsetEnable

```
T_cmnErrorCode mngApi_GetHoldoverOutOfSpecUserDefinedFrequencyOffsetEnable( T_osBool *enable );
```

DESCRIPTION

Get holdover out of specification user defined frequency offset enable.

ARGUMENTS

INPUTS

None

OUTPUT

Enable – E_osTrue - use user defined frequency offset when in out of spec holdover, E_osFalse - use computed holdover value.

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.1.3. mngApi_SetHoldoverOutOfSpecUserDefinedFrequencyOffsetEnable

```
T_cmnErrorCode mngApi_SetHoldoverOutOfSpecUserDefinedFrequencyOffsetEnable( T_osBool enable );
```

DESCRIPTION

Set holdover out of specification user defined frequency offset enable.

ARGUMENTS

INPUTS

Enable – E_osTrue - use user defined frequency offset when in out of spec holdover, E_osFalse - use computed holdover value.

OUTPUT

None

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.1.4. mngApi_GetHoldoverOutOfSpecUserDefinedFrequencyOffsetPpb

```
T_cmnErrorCode mngApi_GetHoldoverOutOfSpecUserDefinedFrequencyOffsetPpb( T_osDouble *offsetPpb );
```

DESCRIPTION

Get holdover out of spec user defined frequency offset.

ARGUMENTS

INPUTS

None

OUTPUT

offsetPpb – frequency offset in parts per billion, 10^9

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.1.5. mngApi_SetHoldoverOutOfSpecUserDefinedFrequencyOffsetPpb

```
T_cmnErrorCode mngApi_SetHoldoverOutOfSpecUserDefinedFrequencyOffsetPpb( T_osDouble offsetPpb );
```

DESCRIPTION

Set holdover out of spec user defined frequency offset.

ARGUMENTS

INPUTS

offsetPpb – frequency offset in parts per billion, 10^9

OUTPUT

None

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.1.6. mngApi_GetHoldoverTimeout

```
T_cmnErrorCode mngApi_GetHoldoverTimeout( T_osUint32 *timeout );
```

DESCRIPTION

Get the holdover timeout.

ARGUMENTS

INPUTS

None

OUTPUT

timeout – holdover timeout in seconds

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.1.7. mngApi_SetHoldoverTimeout

```
T_cmnErrorCode mngApi_SetHoldoverTimeout( T_osUint32 const timeout );
```

DESCRIPTION

Get the holdover timeout.

ARGUMENTS

INPUTS

timeout – holdover timeout in seconds

OUTPUT

None

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.1.8. mngApi_GetHoldoverTimerValue

```
T_cmnErrorCode mngApi_GetHoldoverTimerValue( T_osUint32 *timeRemainingSeconds );
```

DESCRIPTION

Get the holdover timer remaining time.

ARGUMENTS

INPUTS

None

OUTPUT

timeRemainingSeconds – holdover timer remaining time in seconds, 0 means timer is not running.

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.1.9. mngApi_GetHoldoverType

```
T_cmnErrorCode mngApi_GetHoldoverType( T_srvLoHoldoverType *holdoverType );
```

DESCRIPTION

Get the holdover type.

ARGUMENTS

INPUTS

None

OUTPUT

holdoverType – holdover type (software/hardware).

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.1.10. mngApi_SetHoldoverType

```
T_cmnErrorCode mngApi_SetHoldoverType( T_srvLoHoldoverType const holdoverType );
```

DESCRIPTION

Set the holdover type.

ARGUMENTS

INPUTS

holdoverType – holdover type (software/hardware)

OUTPUT

None

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.1.11. mngApi_ForceLoStateHoldover

```
T_cmnErrorCode mngApi_ForceLoStateHoldover( void );
```

DESCRIPTION

Force the LO State Machine into Holdover state.

ARGUMENTS

INPUTS

None

OUTPUT

None

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.1.12. mngApi_ClearForceLoStateHoldover

```
T_cmnErrorCode mngApi_ClearForceLoStateHoldover( void );
```

DESCRIPTION

Clear Force Holdover state.

ARGUMENTS

INPUTS

None

OUTPUT

None

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.2 G.8273.2 Physical Layer Assistance

3.2.1. mngApi_GetPhysicalClockCategory

```
T_cmnErrorCode mngApi_GetPhysicalClockCategory( T_srvPhysicalClockCategory *physicalClockCategory );
```

DESCRIPTION

Get physical layer clock category.

ARGUMENTS

INPUTS

None

OUTPUT

physicalClockCategory – physical layer clock category

RETURN

E_cmnErrorCode_OK – On success
E_cmnErrorCode_NotAccepted – Otherwise

3.2.2. mngApi_SetPhysicalClockCategory

```
T_cmnErrorCode mngApi_SetPhysicalClockCategory( T_srvPhysicalClockCategory physicalClockCategory );
```

DESCRIPTION

Set physical layer clock category.

ARGUMENTS

INPUTS
physicalClockCategory – physical layer clock category

OUTPUT
None

RETURN

E_cmnErrorCode_OK – On success
E_cmnErrorCode_NotAccepted – Otherwise

3.2.3. mngApi_GetPhysicalClockCategoryThreshold

```
T_cmnErrorCode mngApi_GetPhysicalClockCategoryThreshold(
                    T_srvPhysicalClockCategory *physicalClockCategoryThreshold );
```

DESCRIPTION

Get physical layer clock category threshold.

ARGUMENTS

INPUTS
None

OUTPUT
physicalClockCategory – physical layer clock category threshold

RETURN

E_cmnErrorCode_OK – On success
E_cmnErrorCode_NotAccepted – Otherwise

3.2.4. mngApi_SetPhysicalClockCategoryThreshold

```
T_cmnErrorCode mngApi_SetPhysicalClockCategoryThreshold (
                      T_srvPhysicalClockCategory phyClockCategoryThreshold );
```

DESCRIPTION

Set physical layer clock category.

ARGUMENTS

INPUTS

physicalClockCategory – physical layer clock category

OUTPUT

None

RETURN

E_cmnErrorCode_OK – On success
E_cmnErrorCode_NotAccepted – Otherwise

3.2.5. mngApi_GetPhysicalPllWaitToRestoreTimeoutValue

```
T_cmnErrorCode mngApi_GetPhysicalPllWaitToRestoreTimeoutValue(
                     T_osUint16 *physicalPllWaitToRestoreTimeoutValue );
```

DESCRIPTION

Get physical PLL wait to restore timeout value.

ARGUMENTS

INPUTS

None

OUTPUT

physicalPllWaitToRestoreTimeoutValue – physical PLL wait to restore timeout value

RETURN

E_cmnErrorCode_OK – On success
E_cmnErrorCode_NotAccepted – Otherwise

3.2.6. mngApi_SetPhysicalPllWaitToRestoreTimeoutValue

```
T_cmnErrorCode mngApi_SetPhysicalPllWaitToRestoreTimeoutValue(
                     T_osUint16 physicalPllWaitToRestoreTimeoutValue );
```

DESCRIPTION

Set physical PLL wait to restore timeout value.

ARGUMENTS

INPUTS

physicalPllWaitToRestoreTimeoutValue – physical PLL wait to restore timeout value

OUTPUT

None

RETURN

E_cmnErrorCode_OK – On success
E_cmnErrorCode_NotAccepted – Otherwise

3.3 GNSS

3.3.1. mngApi_GetGnssClockCategory

```
T_cmnErrorCode mngApi_GetGnssClockCategory( T_gnssClockCategory *gnssClockCategory );
```

DESCRIPTION

Get GNSS clock category.

ARGUMENTS

INPUTS

None

OUTPUT

gnssClockCategory – GNSS clock category

RETURN

E_cmnErrorCode_OK – On success
E_cmnErrorCode_NotAccepted – Otherwise

3.3.2. mngApi_SetGnssClockCategory

```
T_cmnErrorCode mngApi_SetGnssClockCategory( T_gnssClockCategory gnssClockCategory );
```

DESCRIPTION

Set GNSS clock category.

ARGUMENTS

INPUTS

gnssClockCategory – GNSS clock category

OUTPUT

None

RETURN

E_cmnErrorCode_OK – On success
E_cmnErrorCode_NotAccepted – Otherwise

3.3.3. mngApi_GetGnssClockCategoryThreshold

```
T_cmnErrorCode mngApi_GetGnssClockCategoryThreshold(
                      T_gnssClockCategory *gnssClockCategoryThreshold );
```

DESCRIPTION

Get GNSS clock category threshold.

ARGUMENTS

INPUTS

None

OUTPUT

gnssClockCategory – GNSS clock category threshold

RETURN

E_cmnErrorCode_OK – On success
E_cmnErrorCode_NotAccepted – Otherwise

3.3.4. mngApi_SetGnssClockCategoryThreshold

```
T_cmnErrorCode mngApi_SetGnssClockCategoryThreshold(
                      T_gnssClockCategory gnssClockCategoryThreshold );
```

DESCRIPTION

Set GNSS clock category threshold.

ARGUMENTS

INPUTS

gnssClockCategory – GNSS clock category threshold

OUTPUT

None

RETURN

E_cmnErrorCode_OK – On success
E_cmnErrorCode_NotAccepted – Otherwise

3.4 Message Log

3.4.1. mngApi_OpenSyslog

```
T_cmnErrorCode mngApi_OpenSyslog( T_osChar const *syslogIpAddress,
                                  T_osUint16 udpPort,
                                  T_cmnMessageLevelRegister messageLevelMask );
```

DESCRIPTION

Open syslog socket with IPv4 address and udpPort.

ARGUMENTS

INPUTS

syslogIpAddress – pointer to string representation of IPv4 address, ex. "123.0.0.1"
udpPort – UDP port
messageLevelMask – message level bit field mask, see 2.5

OUTPUT

None

RETURN

E_cmnErrorCode_OK – On success
E_cmnErrorCode_NotAccepted – Otherwise

3.4.2. mngApi_CloseSyslog

```
T_cmnErrorCode mngApi_CloseSyslog( void );
```

DESCRIPTION

Close syslog socket.

ARGUMENTS

INPUTS

None

OUTPUT

None

RETURN

E_cmnErrorCode_OK – Always

3.4.3. mngApi_GetListOfMessageLogs

```
T_cmnErrorCode mngApi_GetListOfMessageLogs( T_cmnLogDescription *messageLogDescriptors,
                                            T_osUint8 *numberOfMessageLogs );
```

DESCRIPTION

Get array of message logs.
Use mngApi_GetNumberOfMessageLogs to get number of message logs to allocate size of array.

ARGUMENTS

INPUTS

messageLogDescriptors – pointer to array of, see 2.3
numberOfMessageLogs – maximum size of the array messageLogDescriptor points to

OUTPUT

numberOfMessageLogs – the number of message logs returned.

RETURN

E_cmnErrorCode_OK – On success
E_cmnErrorCode_NotAccepted – Otherwise

3.4.4. mngApi_GetNumberOfMessageLogs

```
T_cmnErrorCode mngApi_GetNumberOfMessageLogs( T_osUint8 *numberOfMessageLogs );
```

DESCRIPTION

Get number of message logs. Intended to be used to size the T_cmnLogDescription array for mngApi_GetListOfMessageLogs.

ARGUMENTS

INPUTS

None

OUTPUT

numberOfMessageLogs – the number of active message logs

RETURN

E_cmnErrorCode_OK – On success
E_cmnErrorCode_NotAccepted – Otherwise

3.4.5. mngApi_CreateCallbackMessageLog

```
T_cmnErrorCode mngApi_CreateCallbackMessageLog (T_CallbackFunctionPointer userCallback,
                                                T_cmnMessageLevelRegister const messageLevels,
                                                T_cmnLogId  *messageLogId );
```

DESCRIPTION

Create a callback message log. Calls the user registered callback function with the message log string.

ARGUMENTS

INPUTS

userCallback – callback function of type T_CallbackFunctionPointer
messageLevels – message level bit field mask, see 2.5

OUTPUT

messageLogId – identifier for created message log

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.4.6. mngApi_CreateFileMessageLog

```
T_cmnErrorCode mngApi_CreateFileMessageLog( T_osChar const *fullyQualifiedFilename,
                                            T_cmnMessageLevelRegister const messageLevels,
                                            T_osBool const purge,
                                            T_osUint32 const maxSize,
                                            T_osUint8 const archives,
                                            T_cmnLogId  *messageLogId );
```

DESCRIPTION

Open a message log.

ARGUMENTS

INPUTS

fullyQualifiedFilename – maximum length, See CMN_MAX_FULL_LOG_FILE_NAME_LIMIT.
ex. "./createTest1.txt"
messageLevels – message level bit field mask, see 2.5
purge – 0 append to existing file, 1 delete existing files
maxSize – size in bytes, maximum file size before archiving
archive – maximum number of archives

OUTPUT

messageLogId – identifier for created message log

RETURN

E_cmnErrorCode_OK – On success
E_cmnErrorCode_NotAccepted – Otherwise

3.4.7. mngApi_DeleteMessageLog

```
T_cmnErrorCode mngApi_DeleteMessageLog( T_cmnLogId messageLogId );
```

DESCRIPTION

Delete a message log.

ARGUMENTS

INPUTS

messageLogId – identifier for created message log

OUTPUT

None

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.4.8. mngApi_GetMessageLogLevel

```
T_cmnErrorCode mngApi_GetMessageLogLevel( T_cmnLogId messageLogId,
                                          T_cmnMessageLevelRegister *messageLevelMask );
```

DESCRIPTION

Get message log level.

ARGUMENTS

INPUTS

messageLogId – identifier for message log

OUTPUT

messageLevelMask – message level bit field mask, see 2.5

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.4.9. mngApi_SetMessageLogLevel

```
T_cmnErrorCode mngApi_SetMessageLogLevel( T_cmnLogId messageLogId,
                                          T_cmnMessageLevelRegister messageLevelMask );
```

DESCRIPTION

Set message log level.

ARGUMENTS

INPUTS

messageLogId – identifier for message log.
messageLevelMask – message level bit field mask, see 2.5

OUTPUT

None

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.4.10. mngApi_GetStdoutMessageLogLevel

```
T_cmnErrorCode mngApi_GetStdoutMessageLogLevel( T_cmnMessageLevelRegister *messageLevelMask );
```

DESCRIPTION

Get STDOUT message log level.

ARGUMENTS

INPUTS

None

OUTPUT

messageLevelMask – message level bit field mask, see 2.5

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.4.11. mngApi_SetStdoutMessageLogLevel

```
T_cmnErrorCode mngApi_SetStdoutMessageLogLevel( T_cmnMessageLevelRegister messageLevelMask );
```

DESCRIPTION

Set STDOUT message log level.

ARGUMENTS

INPUTS

messageLevelMask – message level bit field mask, see 2.5

OUTPUT

None

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.4.12. mngApi_GetSyslogMessageLogLevel

```
T_cmnErrorCode mngApi_GetSyslogMessageLogLevel( T_cmnMessageLevelRegister *messageLevelMask );
```

DESCRIPTION

Get syslog message log level.

ARGUMENTS

INPUTS

None

OUTPUT

messageLevelMask – message level bit field mask, see 2.5

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.4.13. mngApi_SetSyslogMessageLogLevel

```
T_cmnErrorCode mngApi_SetSyslogMessageLogLevel( T_cmnMessageLevelRegister messageLevelMask );
```

DESCRIPTION

Set syslog message log level.

ARGUMENTS

INPUTS

messageLevelMask – message level bit field mask, see 2.5

OUTPUT

None

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.5 Miscellaneous

3.5.1. mngApi_GetCurrentReferenceMaster

```
T_cmnErrorCode mngApi_GetCurrentReferenceMaster( T_ieee1588PortIdentity *portIdentity );
```

DESCRIPTION

Get the current selected reference master.

ARGUMENTS

INPUTS

None

OUTPUT

portIdentity – current LO state reference master

RETURN

E_cmnErrorCode_OK – On success
E_cmnErrorCode_NotConfigured – if no LO state reference master was configured
E_cmnErrorCode_ResponseTimeout – if no response received within response timeout period

3.5.2. mngApi_GetEpochTimeSeconds

```
T_cmnErrorCode mngApi_GetEpochTimeSeconds( T_osUint64 *seconds )
```

DESCRIPTION

Get the current seconds portion of the PTP DPLL Time of Day (ToD) counter.

ARGUMENTS

INPUTS

None

OUTPUT

seconds – PTP DPLL ToD counter seconds portion value

RETURN

E_cmnErrorCode_OK – On success
E_cmnErrorCode_ResponseTimeout – if no response received within response timeout period

3.5.3. mngApi_GetLoStatus

```
T_cmnErrorCode mngApi_GetLoStatus( T_mngApiLoStatus *currentLoStatus );
```

DESCRIPTION

Get the LO state machine status.

ARGUMENTS

INPUTS

None

OUTPUT

currentLoStatus – pointer to T_mngApiLoStatus structure

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.5.4. mngApi_GetServoMode

```
T_cmnErrorCode mngApi_GetServoMode( T_mngApiServoMode *servoMode );
```

DESCRIPTION

Get servo mode.
Frequency Reference tracker will return E_mngApiServoMode_Frequency, all other reference trackers return E_mngApiServoMode_Time.

ARGUMENTS

INPUTS

None

OUTPUT

servoMode – E_mngApiServoMode_Time or E_mngApiServoMode_Frequency

RETURN

E_cmnErrorCode_OK – On success
E_cmnErrorCode_NotAccepted – Otherwise

3.5.5. mngApi_GetSoftwareVersion

```
T_cmnErrorCode mngApi_GetSoftwareVersion( char const **releaseId,
                                          char const **commitId );
```

DESCRIPTION

Get the software release ID and commit ID strings.
The CMN_RELEASE_ID and CMN_COMMIT_ID are defined in cmnVersionId.h.

ARGUMENTS

INPUTS

None

OUTPUT

releaseId – pointer to a release ID string, xx.yy.zz
commitId – pointer to a Git commit ID

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.5.6. mngApi_GetUnqualifiedTimeout

```
T_cmnErrorCode mngApi_GetUnqualifiedTimeout( T_osUint32 *timeout );
```

DESCRIPTION

Get the holdover unqualified timeout.
Matches the JSON configuration “unqualifiedTimeoutSeconds” holdover value.

ARGUMENTS

INPUTS

None

OUTPUT

timeout – unqualified timeout in seconds

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.5.7. mngApi_GetUnqualifiedTimerValue

```
T_cmnErrorCode mngApi_GetUnqualifiedTimerValue( T_osUint32 *timeRemainingSeconds );
```

DESCRIPTION

Get the holdover timer remaining time.

ARGUMENTS

INPUTS

None

OUTPUT

timeRemainingSeconds – holdover timer remaining time in seconds, 0 means timer is not running.

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.5.8. mngApi_SetEpochTimeSeconds

```
T_cmnErrorCode mngApi_SetEpochTimeSeconds( T_osUint64 const seconds );
```

DESCRIPTION

Set seconds portion of the PTP DPLL Time of Day (ToD) counter, the nanoseconds will be unchanged.

- Should only be called when there is no PTP master available
- Frequency lock is maintained while the seconds is updated because the nanoseconds portion is unchanged
- There should be at least 10 seconds between calls to this function

When using AdaptiveTimeAssist and PTP is not present, this function will set the seconds part of the system ToD. The nanoseconds part will come from the external 1 Hz edge. The time stampers will then be aligned to the system ToD.
When using AdaptiveTimeAssist and PTP is present, PTP will be the source of the system ToD.
The GNSS measurement channel is controlled by the AdaptiveTimeAssist tracker in pcm4l and should not be modified outside of pcm4l.

ARGUMENTS

INPUTS

seconds – Epoch time seconds only, nanoseconds will be unchanged

OUTPUT

None

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.5.9. mngApi_SetUnqualifiedTimeout

```
T_cmnErrorCode mngApi_SetUnqualifiedTimeout( T_osUint32 const timeout );
```

DESCRIPTION

Set the holdover unqualified timeout.
Matches the JSON configuration “unqualifiedTimeoutSeconds” holdover value.

ARGUMENTS

INPUTS

timeout – unqualified timeout in seconds

OUTPUT

None

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.5.10. mngApi_GetGnssStatus

```
T_cmnErrorCode mngApi_GetGnssStatus( T_mngApiGnssStatus *currentGnssStatus );
```

DESCRIPTION

Get the GNSS state machine status.

ARGUMENTS

INPUTS

None

OUTPUT

currentGnssStatus – pointer to T_mngApiGnssStatus structure.
Transient state E_gnssState_WaitFirstTodRead reports as E_gnssState_Unqualified.

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.6 Reference Trackers

3.6.1. mngApi_ReferenceTracker_GetCount

```
T_cmnErrorCode mngApi_ReferenceTracker_GetCount( T_osUint8 *numberOfReferenceTrackers );
```

DESCRIPTION

Get number of reference trackers.

ARGUMENTS

INPUTS

None

OUTPUT

numberOfReferenceTrackers – the number of reference trackers

RETURN

E_cmnErrorCode_OK – On success
E_cmnErrorCode_ResponseTimeout – if no response received within response timeout period

3.6.2. mngApi_ReferenceTracker_GetList

```
T_cmnErrorCode mngApi_ReferenceTracker_GetList( T_cmnReferenceTracker *referenceTrackerList,
                                                T_osUint8 *numberOfReferenceTrackers );
```

DESCRIPTION

Get list of reference trackers. Use mngApi_GetNumberOfReferenceTrackers to get number to allocate size of array.

ARGUMENTS

INPUTS

referenceTrackerList – pointer to array of T_cmnReferenceTracker
numberOfReferenceTrackers – the maximum size of the array that referenceTrackerList is pointing to.

OUTPUT

numberOfReferenceTrackers – the number of message logs in the list

RETURN

E_cmnErrorCode_OK – On success
E_cmnErrorCode_NotAccepted – Otherwise

3.6.3. mngApi_ReferenceTracker_GetDownlinkPacketRate (deprecated)

```
T_cmnErrorCode mngApi_ReferenceTracker_GetDownlinkPacketRate(
                    T_osUint16 stackInstNumber,
                    T_srvPacketRate *packetsPerSecond );
```

DESCRIPTION

Get the downlink (sync) packet rate.

ARGUMENTS

INPUTS

stackInstNumber – stack instance number

OUTPUT

packetsPerSecond – packet rate

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.6.4. mngApi_RT_GetDownlinkPacketRate

```
T_cmnErrorCode mngApi_RT_GetDownlinkPacketRate(
                    T_osUint16 stackInstNumber,
                    T_osUint16 trackerInstNumber,
                    T_srvPacketRate *packetsPerSecond );
```

DESCRIPTION

Get the downlink (sync) packet rate.

ARGUMENTS

INPUTS

stackInstNumber – stack instance number
trackerInstNumber – tracker instance number

OUTPUT

packetsPerSecond – packet rate

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.6.5. mngApi_ReferenceTracker_GetUplinkPacketRate (deprecated)

```
T_cmnErrorCode mngApi_ReferenceTracker_GetUplinkPacketRate(
                    T_osUint16 stackInstNumber,
                    T_srvPacketRate *packetsPerSecond );
```

DESCRIPTION

Get the uplink (delay request) packet rate.

ARGUMENTS

INPUTS

stackInstNumber – stack instance number

OUTPUT

packetsPerSecond – packet rate

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.6.6. mngApi_RT_GetUplinkPacketRate

```
T_cmnErrorCode mngApi_RT_GetUplinkPacketRate(
                    T_osUint16 stackInstNumber,
                    T_osUint16 trackerInstNumber,
                    T_srvPacketRate *packetsPerSecond );
```

DESCRIPTION

Get the uplink (delay request) packet rate.

ARGUMENTS

INPUTS

stackInstNumber – stack instance number

trackerInstNumber – tracker instance number

OUTPUT

packetsPerSecond – packet rate

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.6.7. mngApi_ReferenceTracker_GetFloorDelayEstimateSeconds (deprecated)

```
T_cmnErrorCode mngApi_ReferenceTracker_GetFloorDelayEstimateSeconds(
                    T_osUint16 stackInstNumber,
                    T_osDouble *floorDelayEstimate );
```

DESCRIPTION

Get the floor delay estimate, units in seconds.

ARGUMENTS

INPUTS

stackInstNumber – stack instance number

OUTPUT

floorDelayEstimate – floor delay estimate

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.6.8. mngApi_RT_GetFloorDelayEstimateSeconds

```
T_cmnErrorCode mngApi_RT_GetFloorDelayEstimateSeconds(
                    T_osUint16 stackInstNumber,
                    T_osUint16 trackerInstNumber,
                    T_osDouble *floorDelayEstimate );
```

DESCRIPTION

Get the floor delay estimate, units in seconds.

ARGUMENTS

INPUTS

stackInstNumber – stack instance number
trackerInstNumber – tracker instance number

OUTPUT

floorDelayEstimate – floor delay estimate

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.6.9. mngApi_ReferenceTracker_SetFloorDelayEstimateSeconds (deprecated)

```
T_cmnErrorCode mngApi_ReferenceTracker_SetFloorDelayEstimateSeconds(
                    T_osUint16 stackInstNumber,
                    T_osDouble *floorDelayEstimate );
```

DESCRIPTION

Set the floor delay estimate, units in seconds.

ARGUMENTS

INPUTS

stackInstNumber – stack instance number
floorDelayEstimate – floor delay estimate

OUTPUT

None

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.6.10. mngApi_RT_SetFloorDelayEstimateSeconds

```
T_cmnErrorCode mngApi_RT_SetFloorDelayEstimateSeconds(
                    T_osUint16 stackInstNumber,
                    T_osUint16 trackerInstNumber,
                    T_osDouble *floorDelayEstimate );
```

DESCRIPTION

Set the floor delay estimate, units in seconds.

ARGUMENTS

INPUTS

stackInstNumber – stack instance number
trackerInstNumber – tracker instance number
floorDelayEstimate – floor delay estimate

OUTPUT

None

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.6.11. mngApi_ReferenceTracker_GetHighPrecisionFrequencyCorrectionTime (deprecated)

```
T_cmnErrorCode mngApi_ReferenceTracker_GetHighPrecisionFrequencyCorrectionTime(
                    T_osUint16 stackInstNumber,
                    T_osDouble *correctionTimeMinutes );
```

DESCRIPTION

Get the high precision frequency correction time, units in minutes.
This parameter is the value set by the JSON configuration file parameter “highPrecisionFrequencyCorrectionTimeMinutes”.

ARGUMENTS

INPUTS

stackInstNumber – stack instance number

OUTPUT

correctionTimeMinutes – time in minutes servo will gather timestamps

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.6.12. mngApi_RT_GetHighPrecisionFrequencyCorrectionTime

```
T_cmnErrorCode mngApi_RT_GetHighPrecisionFrequencyCorrectionTime(
                    T_osUint16 stackInstNumber,
                    T_osUint16 trackerInstNumber,
                    T_osDouble *correctionTimeMinutes );
```

DESCRIPTION

Get the high precision frequency correction time, units in minutes.
This parameter is the value set by the JSON configuration file parameter "highPrecisionFrequencyCorrectionTimeMinutes".

ARGUMENTS

INPUTS

stackInstNumber – stack instance number
trackerInstNumber – tracker instance number

OUTPUT

correctionTimeMinutes – time in minutes servo will gather timestamps

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.6.13. mngApi_ReferenceTracker_SetHighPrecisionFrequencyCorrectionTime (deprecated)

```
T_cmnErrorCode mngApi_ReferenceTracker_SetHighPrecisionFrequencyCorrectionTime(
                    T_osUint16 stackInstNumber,
                    T_osDouble *correctionTimeMinutes );
```

DESCRIPTION

Set the high precision frequency correction time, units in minutes.
This parameter is the value set by the JSON configuration file parameter "highPrecisionFrequencyCorrectionTimeMinutes".

For an adaptive time reference tracker, a high precision frequency and time estimation and correction will be performed before entering time tracking mode. This parameter determines how long the high precision frequency and time estimation takes. The longer it takes, the more accurate the correction will be. With a good network condition, in other words, low PDV, the value of this parameter could be small, for example, 2. For a large PDV condition, the value should be set to a bigger value, for example, 4 or 8, or even bigger.

ARGUMENTS

INPUTS

stackInstNumber – stack instance number
correctionTimeMinutes – time in minutes servo will gather timestamps

OUTPUT

None

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.6.14. mngApi_RT_SetHighPrecisionFrequencyCorrectionTime

```
T_cmnErrorCode mngApi_RT_SetHighPrecisionFrequencyCorrectionTime(
                    T_osUint16 stackInstNumber,
                    T_osUint16 trackerInstNumber,
                    T_osDouble *correctionTimeMinutes );
```

DESCRIPTION

Set the high precision frequency correction time, units in minutes.
This parameter is the value set by the JSON configuration file parameter "highPrecisionFrequencyCorrectionTimeMinutes".

For an adaptive time reference tracker, a high precision frequency and time estimation and correction will be performed before entering time tracking mode. This parameter determines how long the high precision frequency and time estimation takes. The longer it takes, the more accurate the correction will be. With a good network condition, in other words, low PDV, the value of this parameter could be small, for example, 2. For a large PDV condition, the value should be set to a bigger value, for example, 4 or 8, or even bigger.

ARGUMENTS

INPUTS

stackInstNumber – stack instance number

trackerInstNumber – tracker instance number
correctionTimeMinutes – time in minutes servo will gather timestamps

OUTPUT

None

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.6.15. mngApi_ReferenceTracker_GetOscillatorType (deprecated)

```
T_cmnErrorCode mngApi_ReferenceTracker_GetOscillatorType(
                    T_osUint16 stackInstNumber,
                    T_srvOscillatorType *oscillatorType );
```

DESCRIPTION

Get the oscillator type.

ARGUMENTS

INPUTS

stackInstNumber – stack instance number

OUTPUT

oscillatorType – oscillator type, see T_srvOscillatorType

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.6.16. mngApi_RT_GetOscillatorType

```
T_cmnErrorCode mngApi_RT_GetOscillatorType(
                    T_osUint16 stackInstNumber,
                    T_osUint16 trackerInstNumber,
                    T_srvOscillatorType *oscillatorType );
```

DESCRIPTION

Get the oscillator type.

ARGUMENTS

INPUTS

stackInstNumber – stack instance number
trackerInstNumber – tracker instance number

OUTPUT

oscillatorType – oscillator type, see T_srvOscillatorType

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.6.17. mngApi_ReferenceTracker_SetOscillatorType (deprecated)

```
T_cmnErrorCode mngApi_ReferenceTracker_SetOscillatorType(
                    T_osUint16 stackInstNumber,
                    T_srvOscillatorType *oscillatorType );
```

DESCRIPTION

Set the oscillator type.

ARGUMENTS

INPUTS

stackInstNumber – stack instance number
oscillatorType – oscillator type, see T_srvOscillatorType

OUTPUT

None

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.6.18. mngApi_RT_SetOscillatorType

```
T_cmnErrorCode mngApi_RT_SetOscillatorType(
                    T_osUint16 stackInstNumber,
                    T_osUint16 trackerInstNumber,
                    T_srvOscillatorType *oscillatorType );
```

DESCRIPTION

Set the oscillator type.

ARGUMENTS

INPUTS

stackInstNumber – stack instance number

trackerInstNumber – tracker instance number

oscillatorType – oscillator type, see T_srvOscillatorType

OUTPUT

None

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.6.19. mngApi_ReferenceTracker_GetPdvThreshold (deprecated)

```
T_cmnErrorCode mngApi_ReferenceTracker_GetPdvThreshold(
                    T_osUint16 stackInstNumber,
                    T_srvPdvValues *pdvThreshold );
```

DESCRIPTION

Get the PDV threshold.
This parameter is used to determine PTSF unusable based on the log variance of the PDV. If the log variance exceeds the PDV threshold, PTSF unusable is set.

ARGUMENTS

INPUTS

stackInstNumber – stack instance number

OUTPUT

pdvThreshold – See T_srvPdvValues, log variance range is {-100 - 0}

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.6.20. mngApi_RT_GetPdvThreshold

```
T_cmnErrorCode mngApi_RT_GetPdvThreshold(
                    T_osUint16 stackInstNumber,
                    T_osUint16 trackerInstNumber,
                    T_srvPdvValues *pdvThreshold );
```

DESCRIPTION

Get the PDV threshold.
This parameter is used to determine PTSF unusable based on the log variance of the PDV. If the log variance exceeds the PDV threshold, PTSF unusable is set.

ARGUMENTS

INPUTS

stackInstNumber – stack instance number

trackerInstNumber – tracker instance number

OUTPUT

pdvThreshold – See T_srvPdvValues, log variance range is {-100 – 0}

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.6.21. mngApi_ReferenceTracker_SetPdvThreshold (deprecated)

```
T_cmnErrorCode mngApi_ReferenceTracker_SetPdvThreshold(
                    T_osUint16 stackInstNumber,
                    T_srvPdvValues *pdvThreshold );
```

DESCRIPTION

Set the PDV threshold.
This parameter is used to determine PTSF unusable based on the log variance of the PDV. If the log variance exceeds the PDV threshold, PTSF unusable is set.

ARGUMENTS

INPUTS

stackInstNumber – stack instance number
pdvThreshold – See T_srvPdvValues, log variance range is {-100 – 0}

OUTPUT

None

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.6.22. mngApi_RT_SetPdvThreshold

```
T_cmnErrorCode mngApi_RT_SetPdvThreshold(
                    T_osUint16 stackInstNumber,
                    T_osUint16 trackerInstNumber,
                    T_srvPdvValues *pdvThreshold );
```

DESCRIPTION

Set the PDV threshold.
This parameter is used to determine PTSF unusable based on the log variance of the PDV. If the log variance exceeds the PDV threshold, PTSF unusable is set.

ARGUMENTS

INPUTS

stackInstNumber – stack instance number

trackerInstNumber – tracker instance number
pdvThreshold – See T_srvPdvValues, log variance range is {-100 – 0}

OUTPUT

None

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.6.23. mngApi_ReferenceTracker_GetPdvThresholdExceededHysteresis (deprecated)

```
T_cmnErrorCode mngApi_ReferenceTracker_GetPdvThresholdExceededHysteresis(
                    T_osUint16 stackInstNumber,
                    T_srvPdvValues *pdvThreshold );
```

DESCRIPTION

Get the PDV threshold hysteresis.
Once the PDV threshold is exceeded and PTSF is declared unusable, the PDV log variance must cross below the PDV Threshold minus this value.

ARGUMENTS

INPUTS

stackInstNumber – stack instance number

OUTPUT

pdvThreshold – See T_srvPdvValues, log variance range is {0 – 10}

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.6.24. mngApi_RT_GetPdvThresholdExceededHysteresis

```
T_cmnErrorCode mngApi_RT_GetPdvThresholdExceededHysteresis(
                    T_osUint16 stackInstNumber,
                    T_osUint16 trackerInstNumber,
                    T_srvPdvValues *pdvThreshold );
```

DESCRIPTION

Get the PDV threshold hysteresis.
Once the PDV threshold is exceeded and PTSF is declared unusable, the PDV log variance must cross below the PDV Threshold minus this value.

ARGUMENTS

INPUTS

stackInstNumber – stack instance number

trackerInstNumber – tracker instance number

OUTPUT

pdvThreshold – See T_srvPdvValues, log variance range is {0 – 10}

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.6.25. mngApi_ReferenceTracker_SetPdvThresholdExceededHysteresis (deprecated)

```
T_cmnErrorCode mngApi_ReferenceTracker_SetPdvThresholdExceededHysteresis(
                    T_osUint16 stackInstNumber,
                    T_srvPdvValues *pdvThreshold );
```

DESCRIPTION

Set the PDV threshold hysteresis.
Once the PDV threshold is exceeded and PTSF is declared unusable, the PDV log variance must cross below the PDV Threshold minus this value.

ARGUMENTS

INPUTS

stackInstNumber – stack instance number

pdvThreshold – See T_srvPdvValues, log variance range is {0 – 10}

OUTPUT

None

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.6.26. mngApi_RT_SetPdvThresholdExceededHysteresis

```
T_cmnErrorCode mngApi_RT_SetPdvThresholdExceededHysteresis(
                    T_osUint16 stackInstNumber,
                    T_osUint16 trackerInstNumber,
                    T_srvPdvValues *pdvThreshold );
```

DESCRIPTION

Set the PDV threshold hysteresis.
Once the PDV threshold is exceeded and PTSF is declared unusable, the PDV log variance must cross below the PDV Threshold minus this value.

ARGUMENTS

INPUTS

stackInstNumber – stack instance number

trackerInstNumber – tracker instance number

pdvThreshold – See T_srvPdvValues, log variance range is {0 – 10}

OUTPUT

None

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.6.27. mngApi_ReferenceTracker_GetStationarityMeasure1LowerBound (deprecated)

```
T_cmnErrorCode mngApi_ReferenceTracker_GetStationarityMeasure1LowerBound(
                    T_osUint16 stackInstNumber,
                    T_osDouble *boundary );
```

DESCRIPTION

Get the Stationarity Measure 1 Lower bound value

ARGUMENTS

INPUTS

stackInstNumber – stack instance number

OUTPUT

boundary – {0.00, 1.00}, default 0.60

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.6.28. mngApi_RT_GetStationarityMeasure1LowerBound

```
T_cmnErrorCode mngApi_RT_GetStationarityMeasure1LowerBound(
                    T_osUint16 stackInstNumber,
                    T_osUint16 trackerInstNumber,
                    T_osDouble *boundary );
```

DESCRIPTION

Get the Stationarity Measure 1 Lower bound value

ARGUMENTS

INPUTS

stackInstNumber – stack instance number

trackerInstNumber – tracker instance number

OUTPUT

boundary – {0.00, 1.00}, default 0.60

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.6.29. mngApi_ReferenceTracker_SetStationarityMeasure1LowerBound (deprecated)

```
T_cmnErrorCode mngApi_ReferenceTracker_SetStationarityMeasure1LowerBound(
                    T_osUint16 stackInstNumber,
                    T_osDouble *boundary );
```

DESCRIPTION

Set the Stationarity Measure 1 Lower bound value
The stationarity measure 1 lower and upper are bounds of the ratio of the PDV's second order statistics on the first half observation window and the second half observation window. For ideal stationary case, the stationarity measure 1 should be close to 1.

ARGUMENTS

INPUTS

stackInstNumber – stack instance number

boundary – {0.00, 1.00}, default 0.60

OUTPUT

None

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.6.30. mngApi_RT_SetStationarityMeasure1LowerBound

```
T_cmnErrorCode mngApi_RT_SetStationarityMeasure1LowerBound(
                    T_osUint16 stackInstNumber,
                    T_osUint16 trackerInstNumber,
                    T_osDouble *boundary );
```

DESCRIPTION

Set the Stationarity Measure 1 Lower bound value
The stationarity measure 1 lower and upper are bounds of the ratio of the PDV's second order statistics on the first half observation window and the second half observation window. For ideal stationary case, the stationarity measure 1 should be close to 1.

ARGUMENTS

INPUTS

stackInstNumber – stack instance number

trackerInstNumber – tracker instance number

boundary – {0.00, 1.00}, default 0.60

OUTPUT

None

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.6.31. mngApi_ReferenceTracker_GetStationarityMeasure1UpperBound (deprecated)

```
T_cmnErrorCode mngApi_ReferenceTracker_GetStationarityMeasure1UpperBound(
                    T_osUint16 stackInstNumber,
                    T_osDouble *boundary );
```

DESCRIPTION

Get the Stationarity Measure 1 Upper bound value

ARGUMENTS

INPUTS

stackInstNumber – stack instance number

OUTPUT

boundary – {1.00, 1000.00}, default 1.67

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.6.32. mngApi_RT_GetStationarityMeasure1UpperBound

```
T_cmnErrorCode mngApi_RT_GetStationarityMeasure1UpperBound(
                    T_osUint16 stackInstNumber,
                    T_osUint16 trackerInstNumber,
                    T_osDouble *boundary );
```

DESCRIPTION

Get the Stationarity Measure 1 Upper bound value

ARGUMENTS

INPUTS

stackInstNumber – stack instance number

trackerInstNumber – tracker instance number

OUTPUT

boundary – {1.00, 1000.00}, default 1.67

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.6.33. mngApi_ReferenceTracker_SetStationarityMeasure1UpperBound (deprecated)

```
T_cmnErrorCode mngApi_ReferenceTracker_SetStationarityMeasure1UpperBound(
                    T_osUint16 stackInstNumber,
                    T_osDouble *boundary );
```

DESCRIPTION

Set the Stationarity Measure 1 Upper bound value
The stationarity measure 1 lower and upper are bounds of the ratio of the PDV's second order statistics on the first half observation window and the second half observation window. For ideal stationary case, the stationarity measure 1 should be close to 1.

ARGUMENTS

INPUTS

stackInstNumber – stack instance number

boundary – {1.00, 1000.00}, default 1.67

OUTPUT

None

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.6.34. mngApi_RT_SetStationarityMeasure1UpperBound

```
T_cmnErrorCode mngApi_RT_SetStationarityMeasure1UpperBound(
                    T_osUint16 stackInstNumber,
                    T_osUint16 trackerInstNumber,
                    T_osDouble *boundary );
```

DESCRIPTION

Set the Stationarity Measure 1 Upper bound value
The stationarity measure 1 lower and upper are bounds of the ratio of the PDV's second order statistics on the first half observation window and the second half observation window. For ideal stationary case, the stationarity measure 1 should be close to 1.

ARGUMENTS

INPUTS

stackInstNumber – stack instance number

trackerInstNumber – tracker instance number

boundary – {1.00, 1000.00}, default 1.67

OUTPUT

None

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.6.35. mngApi_ReferenceTracker_GetWillCorrectFrequencyAtFirstSnap (deprecated)

```
T_cmnErrorCode mngApi_ReferenceTracker_GetWillCorrectFrequencyAtFirstSnap(
                    T_osUint16 stackInstNumber,
                    T_osBool *willCorrect );
```

DESCRIPTION

This parameter configures the frequency correction after REA servo performs the first time snap. If the network condition is good, in other words, very little PDV, the accuracy of the first coarse time of day estimation and the LO frequency offset estimation could be good enough and a frequency correction can be performed. On the other hand, if the PDV is large, the initial coarse frequency estimation could have a very large error, thus the frequency correction is preferred to not be made after the first snap

ARGUMENTS

INPUTS

stackInstNumber – stack instance number

OUTPUT

willCorrect – E_osTrue or E_osFalse

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.6.36. mngApi_RT_GetWillCorrectFrequencyAtFirstSnap

```
T_cmnErrorCode mngApi_RT_GetWillCorrectFrequencyAtFirstSnap(
                    T_osUint16 stackInstNumber,
                    T_osUint16 trackerInstNumber,
                    T_osBool *willCorrect );
```

DESCRIPTION

This parameter configures the frequency correction after REA servo performs the first time snap. If the network condition is good, in other words, very little PDV, the accuracy of the first coarse time of day estimation and the LO frequency offset estimation could be good enough and a frequency correction can be performed. On the other hand, if the PDV is large, the initial coarse frequency estimation could have a very large error, thus the frequency correction is preferred to not be made after the first snap.

ARGUMENTS

INPUTS

stackInstNumber – stack instance number

trackerInstNumber – tracker instance number

OUTPUT

willCorrect – E_osTrue or E_osFalse

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.6.37. mngApi_ReferenceTracker_SetWillCorrectFrequencyAtFirstSnap (deprecated)

```
T_cmnErrorCode mngApi_ReferenceTracker_SetWillCorrectFrequencyAtFirstSnap(
                    T_osUint16 stackInstNumber,
                    T_osBool *willCorrect );
```

DESCRIPTION

This parameter configures the frequency correction after REA servo performs the first time snap. If the network condition is good, in other words, very little PDV, the accuracy of the first coarse time of day estimation and the LO frequency offset estimation could be good enough and a frequency correction can be performed. On the other hand, if the PDV is large, the initial coarse frequency estimation could have a very large error, thus the frequency correction is preferred to not be made after the first snap.

ARGUMENTS

INPUTS

stackInstNumber – stack instance number
willCorrect – E_osTrue or E_osFalse

OUTPUT

None

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.6.38. mngApi_RT_SetWillCorrectFrequencyAtFirstSnap

```
T_cmnErrorCode mngApi_RT_SetWillCorrectFrequencyAtFirstSnap(
                    T_osUint16 stackInstNumber,
                    T_osUint16 trackerInstNumber,
                    T_osBool *willCorrect );
```

DESCRIPTION

This parameter configures the frequency correction after REA servo performs the first time snap. If the network condition is good, in other words, very little PDV, the accuracy of the first coarse time of day estimation and the LO frequency offset estimation could be good enough and a frequency correction can be performed. On the other hand, if the PDV is large, the initial coarse frequency estimation could have a very large error, thus the frequency correction is preferred to not be made after the first snap.

ARGUMENTS

INPUTS

stackInstNumber – stack instance number
trackerInstNumber – tracker instance number
willCorrect – E_osTrue or E_osFalse

OUTPUT

None

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.6.39. mngApi_ReferenceTracker_GetFfoSlopeLimitPpbPerSecond (deprecated)

```
T_cmnErrorCode mngApi_ReferenceTracker_GetFfoSlopeLimitPpbPerSecond(
                    T_osUint16 stackInstNumber,
                    T_osDouble *ffoSlopeLimitPpbPerSecond );
```

DESCRIPTION

Get the FFO slope limit, measured in parts per billion per second, used during frequency lock and time locked state.

ARGUMENTS

INPUTS

stackInstNumber – stack instance number

OUTPUT

ffoSlopeLimitPpbPerSecond – {0, 100,000}, default -1

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.6.40. mngApi_RT_GetFfoSlopeLimitPpbPerSecond

```
T_cmnErrorCode mngApi_RT_GetFfoSlopeLimitPpbPerSecond(
                    T_osUint16 stackInstNumber,
                    T_osUint16 trackerInstNumber,
                    T_osDouble *ffoSlopeLimitPpbPerSecond );
```

DESCRIPTION

Get the FFO slope limit, measured in parts per billion per second, used during frequency lock and time locked state.

ARGUMENTS

INPUTS

stackInstNumber – stack instance number
trackerInstNumber – tracker instance number

OUTPUT

ffoSlopeLimitPpbPerSecond – {0, 100,000}, default -1

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.6.41. mngApi_ReferenceTracker_SetFfoSlopeLimitPpbPerSecond (deprecated)

```
T_cmnErrorCode mngApi_ReferenceTracker_SetFfoSlopeLimitPpbPerSecond(
                    T_osUint16 stackInstNumber,
                    T_osDouble *ffoSlopeLimitPpbPerSecond );
```

DESCRIPTION

Set the FFO slope limit, measured in parts per billion per second, used during frequency lock and time locked state.

ARGUMENTS

INPUTS

stackInstNumber – stack instance number
ffoSlopeLimitPpbPerSecond – {0, 100,000}, default -1

OUTPUT

None

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

3.6.42. mngApi_RT_SetFfoSlopeLimitPpbPerSecond

```
T_cmnErrorCode mngApi_RT_SetFfoSlopeLimitPpbPerSecond(
                    T_osUint16 stackInstNumber,
                    T_osUint16 trackerInstNumber,
                    T_osDouble *ffoSlopeLimitPpbPerSecond );
```

DESCRIPTION

Set the FFO slope limit, measured in parts per billion per second, used during frequency lock and time locked state.

ARGUMENTS

INPUTS

stackInstNumber – stack instance number
trackerInstNumber – tracker instance number
ffoSlopeLimitPpbPerSecond – {0, 100,000}, default -1

OUTPUT

None

RETURN

E_cmnErrorCode_OK – On success, else another T_cmnErrorCode

4. Revision History

Revision	Date	Description
1.02	Oct 24, 2024	Updated software release number to 4.3.2.417904 from 4.3.2.
1.01	Oct 15, 2024	Supports software release 4.3.2. ▪ Added APIs: • mngApi_GetPhysicalPllWaitToRestoreTimeoutValue • mngApi_SetPhysicalPllWaitToRestoreTimeoutValue • mngApi_ReferenceTracker_GetFfoSlopeLimitPpbPerSecond • mngApi_ReferenceTracker_SetFfoSlopeLimitPpbPerSecond • mngApi_GetEpochTimeSeconds • mngApi_SetEpochTimeSeconds ▪ Removed APIs: • mngApi_GetHoldoverEnable • mngApi_SetHoldoverEnable ▪ Updated definition of T_srvOscillatorType enumeration • New enumeration E_srvDocxo replaced E_srvSyncE ▪ Update function prototypes of Reference tracker management API functions • New additional input parameter trackerInstNumber ▪ Add T_mngApiLoLockStatus and T_srvLoStateId enumeration
1.00	Jun 21, 2021	Initial release. Supports software release 4.1.0.77765.

IMPORTANT NOTICE AND DISCLAIMER

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES (“RENESAS”) PROVIDES TECHNICAL SPECIFICATIONS AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD-PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for developers who are designing with Renesas products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Renesas grants you permission to use these resources only to develop an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third-party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising from your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use of any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Disclaimer Rev.1.01 Jan 2024)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit www.renesas.com/contact-us/.

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.