IPCLOCK

IPC9xxx/17xx/1603
IEEE 1588 v2
Boundary, Slave and Master Clocks
User's Guide

Table of Content

1 Introduction

The IPC9xxx/17xx/1603 utilize IPClock's state-of-the-art IEEE 1588 v2 technology optimized for providing high quality frequency synchronization and Time of Day (ToD) over packet transport networks. Clock synchronization is required by many wireline and wireless applications including 3G, 4G-LTE, Smart Grid, Industrial automation and aerospace and defense. The IPC9xxx/17xx/1603 are application-agnostic, cost effective, standard compliant IEEE 1588 v2 BC/Slave/Master, supporting G.8265.1 and G.8275.1. The IPC9xxx/17xx/1603 are designed for easy field upgrades to support future enhancements as well as future synchronization standards.

2 Functional Block Diagram

The IPC9xxx/17xx/1603 functional block diagram is depicted in Figure 1 below.

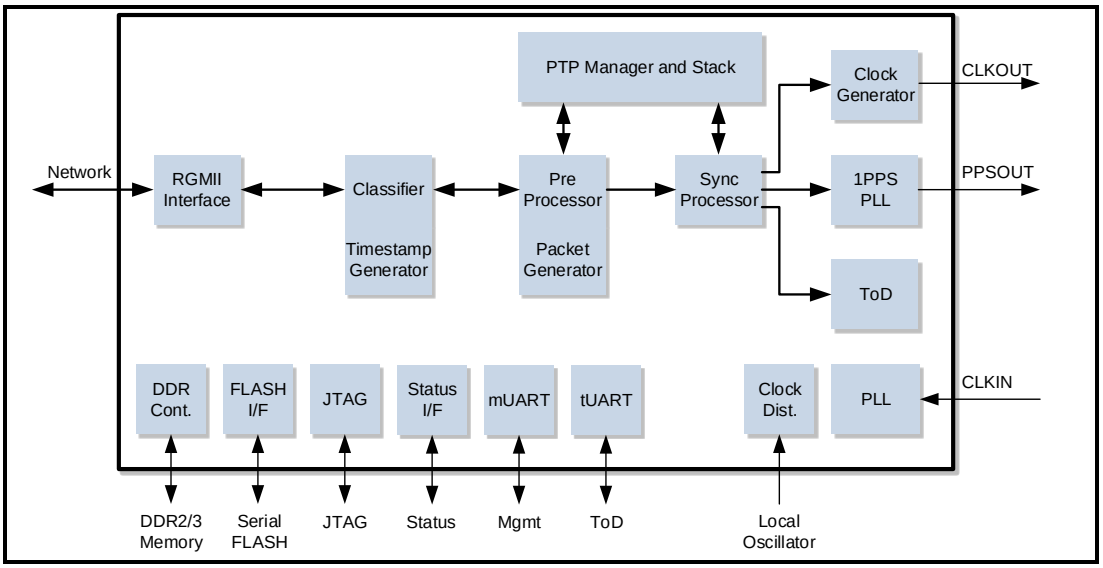

Figure 1: IPC9xxx/17xx 1588 v2 BC/Slave/Master Functional Block Diagram

The IPC9xxx/17xx/1603 can be set to operate as either IEEE 1588 v2 Boundary Clock (IPC9xxx) Master (IPC9xxx/17xx) or Slave. The IEEE 1588 v2 protocol is a bidirectional protocol requiring all ports to transmit and receive 1588 v2 packets. Each packet received its time-stamp by the Timestamp Generator and classified by the Classifier. In case the packet is 1588 v2, packet it is sent to the Pre-Processor along with its timestamp. The Pre-Processor is transferring the received general packets to the PTP Manager and Stack for further processing. In the case of 1588 v2 event packet the Pre-Processor compensate for part of the packet network impairments and prepare the data for the Sync Processor. The Sync Processor is comprised of a suite of algorithms that processes the data and controls the 1PPS PLL or the 3rd party EEC/DCO, the four programmable clock outputs of the Clock Generator, and the ToD. The ToD is communicating with the ToD UART (tUART) utilizing the NMEA protocol for either providing or getting the ToD from a GPS. The 1588 v2 packets are transmitted by the Packet Generator, which is controlled by the PTP Manager and Stack. Each packet transmitted is time-stamped by the Timestamp Generator and this timestamp is either embedded into the packet or sent to the Pre-Processor depending on the packet type and selected mode of operation.

The IPC9xxx/17xx/1603 package includes two parts: mcs file for the FLASH memory and software driver. The mcs file shall be burned into FLASH memory starting at address 0. The software driver is delivered in source code and includes code examples. It enables to control the devices using the APIs as describe in this document. The IPC9xxx/17xx/1603 are designed for standalone operation with or without host CPU. Configuring, controlling, and monitoring of the device can be done by commands through the management UART (mUART). The management UART supports two modes of operations: Terminal and Host.

3 Software Management Driver

3.1 General

The management UART supports two modes of operations: (1) Terminal, and (2) Host.

- Terminal - The Terminal mode can be used for working with terminal software. In this mode the IPC9xxx/17xx/1603 are been controlled directly by the user without the need for software driver. The UART echo is active (the device is transmitting back every character it received). The returned values displayed by the terminal software.
- Host - The Host mode can be used for working with host CPU. In this mode the Software Management Driver enables the user application to control the IPC9xxx/17xx/1603. The UART echo is disabled, the returned values are the Hexadecimal representation of the ASCII characters. For robust management, API messages contain CRC16. CRC is added to end of each messages (string contains name of API with parameters) sending by the Host to target device. CRC length is 4 hexadecimal digits in ASCII. The device adds CRC16 to each response message it returns to Host besides messages contains log data. The receiving side checks the CRC. In case of the mismatch ERROR -4 (target) and ERROR -5 (Host) shall be returned.

The Software Management Driver enables the HOST CPU to control the IPC9xxx/17xx/1603. The driver includes 3 layers: Device Driver, Control Driver and Driver API. The Device Driver and the Control Driver provide the services to all the APIs. The Driver API includes the API specific code. The driver software package includes Host Application Example. The user application shall use the Driver API.

The driver package includes four (4) cpp files. The modules and their file names are presented in Table 1.

Module	File name
Device Driver	.HostMgmt/DeviceDriver/src/IpcDeviceDriver.cpp
Control Driver	.HostMgmt/CtrlDriver/src/IpcCtrlDriver.cpp
Driver API	.HostMgmt/DriverApi/src/IpcDriverApi.cpp
Host Application Example	.HostMgmt/SampleHostApp/src/IpcHostApp.cpp

Table 1: Software Driver Modules and File Names

The driver package includes six (6) h files. The file names are presented in Table 2.

File name
.HostMgmt/DeviceDriver/include/IpcDeviceDriver.h
.HostMgmt/CtrlDriver/include/IpcCtrlDriver.h
.HostMgmt/DriverApi/include/IpcDriverApi.h
.HostMgmt/SampleHostApp/include/IpcHostApp.h
.HostMgmt/include/include/IpcHostMgmtGeneral.h
.HostMgmt/include/include/IpcDef.h

Table 2: Software Driver Modules and File Names

The user implementation shall follow those guidelines:

- The file IpcDeviceDriver.cpp contains the Device Driver using Xilinx UART Lite library. The h-files: xuartlite.h and xuartlite_l.h reference to this software. The user shall use this library or analogs of functions XUartLite_Send(), XUartLite_Recv(), XUartLite_IsSending().
- Including xparameters.h can be ignored.
- Including sys/process.h can be ignored. It uses to get access to function yield(). This function uses in order to improve the transmission of data.
- User shall rewrite the functions DeviceDriver_SendRequestMsg() and DeviceDriver_RecieveResponseMsg() in his environment.

It is recommended to call the device APIs in a rate not exceed one call per sec.

APIs shall be call only after the device finished its initialization process and ready for operation. The device is ready for operation when gState.state!=0 (not Init) as reported by GetIpcPtpState API. Typical start-up process, while calling GetIpcPtpState is as follow:

```
Application load
IPC9000-20-v2.40
CDB4
Applying default parameters
================================
Slave Port BIST Report - Init
--------------------------------
FPGA read                - PASSED
FPGA write & read 1st    - PASSED
FPGA write & read 2nd    - PASSED
PLL-1                    - PASSED
PLL-2                    - PASSED
PLL-3                    - PASSED
PLL-4                    - PASSED
Clock sync init          - PASSED
--------------------------------
Slave Port BIST Init     - PASSED
================================
================================
Master Port BIST Report - Init
--------------------------------
FPGA read                - N/A
FPGA write & read 1st    - N/A
FPGA write & read 2nd    - N/A
PLL-1                    - N/A
PLL-2                    - N/A
PLL-3                    - N/A
PLL-4                    - N/A
Clock sync init          - PASSED
--------------------------------
Master Port BIST Init    - PASSED
================================
00:00:00:01 [#1100] [GetIpcAdminIfcNetMacAddr]: 00:0A:35:01:F4:16
00:00:00:01 [#1100] [GetIpcAdminIfcNetIpAddr]: 192.168.1.22
00:00:00:01 [#1100] [GetIpcAdminIfcNetIpSubnetMask]: 255.255.255.0
00:00:00:01 [#1100] [GetIpcAdminIfcNetIpDefaultGateway]: 192.168.1.1
00:00:00:01 [#1100] [GetIpcPtpDsClockIdentity]: 00:0A:35:FF:FE:01:F4:16
00:00:00:01 [#1100] [GetIpcPtpAdminRole]: 4
GetIpcPtpState
00:00:00:00 [#1100] [GetIpcPtpState]: 9 0 0
result = OK
GetIpcPtpState
00:00:00:05 [#1100] [GetIpcPtpState]: 1 0 0
result = OK
GetIpcPtpState
00:00:00:10 [#1100] [GetIpcPtpState]: 1 0 0
result = OK
00:00:00:13 [#0113] IPC9000 Boundary Clock is ready
00:00:00:14 [#0152] ST2 0-->1
GetIpcPtpState
00:00:00:15 [#1100] [GetIpcPtpState]: 1 1 1
result = OK
```

3.2 Management Modes

The SetIpcAdminMgmtMode enables to control the Management mode of operation.
The following example presents the two modes of operation:

```
GetIpcAdminMgmtMode(0)                  // Get management mode (in Terminal mode)
[GetIpcAdminMgmtMode]: 0                // Echo API name, follow by returned information (0)
result = OK                             // STATUS

GetIpcPtpState(0,0)                     // Get PTP state
[GetIpcPtpState]: 2 4 8                 // Echo API name, follow by returned information (2,4,8)
result = OK                             // STATUS

SetIpcAdminMgmtMode(0,1)                // Set device to Host management mode
$00000000@                              // Message response from the device in hexadecimal
```

```
                                    // Including STATUS (OK) and CRC
                                    // The first 4 hexadecimal digits starting from the $ sign
                                    // are the STATUS (OK), the next 4 hexadecimal digits
                                    // are the CRC

GetIpcPtpState(0,0)                 // Get device PTP state
$0000040802000E@                    // Message response from the device in hexadecimal
                                    // Including STATUS (OK),  and CRC
                                    // The first 4 hexadecimal digits starting from the $ sign
                                    // (0000) are the STATUS (OK), the next 6 hexadecimal
                                    // digits (040802) are the GState structure, the next 4
                                    // hexadecimal digits (000E) are the CRC
```

In Host mode of operation the user should replace the functions below in the file IpcDeviceDriver.cpp with its own functions that have the same functionality:

- int DeviceDriver_SendRequestMsg(void* outputMsg)
- int DeviceDriver_RecieveResponseMsg(void* inputMsg, int* msgLen)

4 Definitions and Types

The following types and definitions are included in IpcApiDef.h file.

4.1 Primitive Data Types

```
typedef int16_t   int16;
typedef int32_t   int32;
typedef int64t    int64;
typedef u_char_t  u_char;
typedef uint16_t  Uint16;
typedef uint32_t  Uint32;
typedef struct    int48t {
        int16_t   msb;        // Most significant bits (high-order bits)
        uint32_t  lsb;        // Least significant bits (low-order bits)
} __attribute__((packed)) int48t;
typedef struct    uint48t {
        uint16_t  msb;        // Most significant bits (high-order bits)
        uint32_t  lsb;        // Least significant bits (low-order bits)
} __attribute__((packed)) uint48t;
typedef int64_t   int64t;
```

4.2 Primitive PTP Data Types

```
typedef bool      Boolean;          // Primitive PTP data types
                                    // typedef enum {FALSE=0, TRUE} Boolean
typedef u_char_t  Enumeration4;     // 4-bit enumerated value
typedef u_char_t  Enumeration8;     // 8-bit enumerated value
typedef uint16_t  Enumeration16;    // 16-bit enumerated value
typedef u_char_t  UInteger4;        // 4-bit enumerated value
typedef int8_t    Integer8;         // 8-bit signed integer
typedef u_char_t  UInteger8;        // 8-bit unsigned integer
typedef int16_t   Integer16;        // 16-bit signed integer
typedef uint16_t  UInteger16;       // 16-bit unsigned integer
typedef int32_t   Integer32;        // 32-bit signed integer
typedef uint32_t  UInteger32;       // 32-bit unsigned integer
typedef uint48t   UInteger48;       // 48-bit unsigned integer
typedef int64t    Integer64;        // 64-bit signed integer
typedef u_char_t  Nibble;           // 4-bit field not interpreted as a number
typedef u_char_t  Octet;            // 8-bit field not interpreted as a number
```

4.3 PTP Data Types

```
typedef struct PTPText32 {
      UInteger8   lengthField;
      Octet       textField[32];
```

```
} PTPText32;
typedef struct PTPText64 {
      UInteger8   lengthField;
      Octet       textField[64];
} PTPText64;
typedef struct PTPText128 {
      UInteger8   lengthField;
      Octet       textField[128];
} PTPText128;
```

4.4 Derived Data Types

4.4.1 General

4.4.1.1 IPC_Uint48

```
typedef struct {
        Uint16 High;
        Uint32 Low;
} IPC_Uint48;
```

4.4.1.2 IPC_STATUS

```
typedef int16 STATUS;
```

4.4.1.3 ClockIdentity

```
#define CLOCK_IDENTITY_LEN (23 +1)
typedef char ClockIdentity_t[CLOCK_IDENTITY_LEN];
```

4.4.1.4 TBistTestResult_t

```
typedef u_char TBistTestResult_t;
/* 0 - N/A                          */
/* 1 - PASSED                       */
/* 2 - FAILED                       */
/* 3 - ALARMED                      */
```

4.4.1.5 MacAddr48

```
#define EUI48_LEN (17 + 1)
typedef char MacAddr48[EUI48_LEN];
```

4.4.1.6 MacAddr

```
#define MAC_ADDR_SIZE 6
typedef struct MacAddr {
    u_char addr[MAC_ADDR_SIZE];
} MacAddr;
```

4.4.1.7 IpAddrV4

```
#define IPV4_LEN (15 + 1)
typedef char IpAddrV4[IPV4_LEN];
```

4.4.1.8 SubnetMaskV4

```
Typedef char SubnetMaskV4[IPV4_LEN];
```

4.4.1.9 IfcNetConf

```
typedef struct IfcNetConf {
      int32             ifcNetNmbr;       /*Interface number 0 only     */
      int32             ifcNetEn;         /*Port enable/disable         */
      Uint16            ifcMacSpeed;      /*Port speed                  */
      MacAddr48         macAddress;       /*MAC address                 */
      IpAddrV4          ipAddr;           /*IP address                  */
```

```
      IpAddrV4          subnetMask;       /*Subnet mask                 */
      IpAddrV4          defaultGw;        /*Default gateway address     */
} IfcNetConf;
```

4.4.1.10 UniAddr

```
// Uniform Address
typedef union UniAddr {
      u_char            pad[sizeof(MacAddr)];
      Uint32            ipAddr;
      MacAddr           macAddr;
} UniAddr;
```

4.4.1.11 IpcEthArpEntry

```
typedef struct IpcEthArpEntry {
      Uint32            ipaddr;
      MacAddr           ethaddr;
      u_char            state;            // 0 - Empty
                                          // 1 - Pending
                                          // 2 - Stable
      u_char            type;             // 0 - Automatic
                                          // 1 - Static
} IpcEthArpEntry;
```

4.4.1.12 TxPathDelay

```
typedef struct TxPathDelay {
      int32             txPathDelay_ns;
      int32             txPathDelay_sub_ns;
} TxPathDelay;
```

4.4.1.13 RxPathDelay

```
typedef struct RxPathDelay {
      int32             rxPathDelay_ns;
      int32             rxPathDelay_sub_ns;
} RxPathDelay;
```

4.4.1.14 ClkOutEn

```
typedef struct {
      Uint32            clkOutEn1;
      Uint32            reserved2;
      Uint32            reserved3;
      Uint32            reserved4;
      Uint32            ppsOut;
} ClkOutEn;
```

4.4.1.15 StartPar

```
typedef struct {
      int32             startMode;
      int32             syncPktRate;
      int32             extDevMode;
      int32             portMapMode;
} StartPar;
```

4.4.1.16 TraceabilityProperties

```
typedef struct TraceabilityProperties {
      int32             timeTraceable;
      int32             frequencyTraceable;
} TraceabilityProperties;
```

4.4.1.17 TimescaleProperties

```
typedef struct TimescaleProperties {
      int16             ptpTimescale;
      int32             timeSource;
} TimescaleProperties;
```

4.4.1.18 UtcProperties

```
typedef struct UtcProperties {
      int16             currentUtcOffset;
      int32             currentUtcOffsetValid;
      int32             leap59;
      int32             leap61;
} UtcProperties;
```

4.4.1.19 PtpBmcModeEx

```
typedef struct {
      int32             msg;
      int32             fourcePrio1_255;
      int32             ignoreUniMasterTable;
      int32             etiUpdateInterval;
      int32             delSel;
      int32             stepsRemovedMax;
      int32             clsPortNumMode;
      int32             IgnoreParent_Ho_HoSpecDis;
      int32             report;
} PtpBmcModeEx;
```

4.4.1.20 NcaPar

```
typedef struct {
      int32             bcPktRateLimitMode=0;
      int32             limit4x128ch=0;
      int32             addDReq1=0;
} NcaPar;
```

4.4.1.21 IpDscp

```
typedef struct {
      u_char            ipDscpPtpEvent;
      u_char            ipDscpPtpGeneral;
} IpDscp;
```

4.4.1.22 IpEsf

```
typedef struct {
      u_char            ipDscpPtpEvent;
      u_char            ipEcnPtpEvent;
      u_char            ipDscpPtpGeneral;
      u_char            ipEcnPtpGeneral;
} IpDsf;
```

4.4.2 Time

4.4.2.1 PtpTimeNs

```
typedef struct {
      IPC_Uint48        sec;
      Uint32            nsec;
} PtpTimeNs;
```

4.4.2.2 GpsTime

```
typedef struct {
      Uint32            gpsWeeks;
```

```
      Uint32            secInLastWeek;
} GpsTime;
```

4.4.2.3 GpsTimeNs

```
typedef struct {
      GpsTime           gpsTime;
      Uint32            nsec;
} GpsTimeNs;
```

4.4.2.4 UtcTime

```
typedef struct {
      Uint16            year;
      Uint16            month;
      Uint16            day;
      Uint16            hour;
      Uint16            min;
      Uint16            sec;
} UtcTime;
```

4.4.2.5 UtcTimeNs

```
typedef struct {
      UtcTime           utcTime;
      Uint32            nsec;
} UtcTimeNs;
```

4.4.2.6 STimeInterval

```
typedef struct STimeInterval {
      int64             scaledNanoseconds;
} TimeInterval_t;
```

4.4.3 Datasets

4.4.3.1 PortIdentity

```
#define CLOCK_IDENTITY_LEN (23 +1)
typedef char ClockIdentity_t[CLOCK_IDENTITY_LEN];

typedef struct PortIdentity {
      ClockIdentity_t               clockIdentity;
      Uint16                        portNumber;
} PortIdentity_t;
```

4.4.3.2 PortIdentityA

```
typedef struct SIeee1588ClockIdentity {
      Uint16                        clockIdentity[4];
} SIeee1588ClockIdentity_t;

typedef struct PortIdentityA {
      SIeee1588ClockIdentity_t      clockIdentity;
      Uint16                        portNumber;
} PortIdentityA;
```

4.4.3.3 ClockQuality

```
typedef struct ClockQuality {
      u_char            clockClass;
      u_char            clockAccuracy;
      Uint16            offsetScaledLogVar;
} ClockQuality_t;
```

4.4.3.4 DefaultDs

```
typedef struct DefaultDs {
      Uint32            twoStepFlag;
      ClockIdentity_t   clockIdentity;
      Uint16            numberPorts;
      ClockQuality_t    clockQuality;
      Uint32            priority1;
      Uint32            priority2;
      Uint32            domainNumber;
      Uint32            slaveOnly;
      u_char            localPriority;      //Applicable for ptpBmcMode=10
} DefaultDs_t;
```

4.4.3.5 CurrentDs

```
typedef struct CurrentDs {
      Uint16            stepsRemoved;
      int64             offsetFromMaster;
      int64             meanPathDelay;
} CurrentDs_t;
```

4.4.3.6 ParentDs

```
typedef struct ParentDs {
      UniAddr           parentUniAddr;
      PortIdentity_t    parentPortIdentity;
      int32             parentStats;
      Uint16            obsParOffsetScaledLogVar;
      int32             obsParClockPhaseChangeRate;
      ClockIdentity_t   grandmasterIdentity;
      ClockQuality_t    grandmasterClockQuality;
      Uint32            grandmasterPriority1;
      Uint32            grandmasterPriority2;
      u_char            LocalPriority;    //Applicable for ptpBmcMode=10
} ParentDs_t;
```

4.4.3.7 TimePropertiesDs

```
typedef struct TimePropertiesDs {
      int16             currentUtcOffset;
      int32             currentUtcOffsetValid;
      int32             leap59;
      int32             leap61;
      int32             timeTraceable;
      int32             frequencyTraceable;
      int32             ptpTimescale;
      Uint32            timeSource;
} TimePropertiesDs_t;
```

4.4.3.8 PortDs

```
typedef struct PortDs {
      PortIdentity_t    portIdentity;
      u_char            portState;
      char              logMinDelayReqInterval;
      TimeInterval_t    peerMeanPathDelay;
      char              logAnnounceInterval;
      Uint16            announceReceiptTimeout;
      char              logSyncInterval;
      u_char            delayMechanism;
      char              logMinPdelayReqInterval;
      u_char            versionNumber;
      u_char            localPriority;    //Applicable for ptpBmcMode=10
      u_char            notSlave;         //Applicable for ptpBmcMode=10
```

```
} PortDS_t

PortDS_t portDs[PORT_INDEX_MAX];

# define PAGE_SIZE = 16
typedef struct {
      PortDS_t          portDs[PAGE_SIZE];
} PortDS_s;

# define PORT_INDEX_MAX_ = 64
typedef struct {
      PortDS_t          portDs[PORT_INDEX_MAX];
} PortDSAll_s;
```

4.4.4 Port

4.4.4.1 PortNums_t

```
#define PORT_INDEX_MAX 64                  // for 64 channels
typedef struct PortNums {
      Uint16 portNums[PORT_INDEX_MAX];
} PortNums_t;
```

4.4.5 SelectedMaster

4.4.5.1 SelectedMasterEx

```
typedef struct SelectedMasterEx {
      ClockIdentity_t   grandmasterIdentity;    // GetIpcPtpDsParent
      PortIdentity_t    parentPortIdentity;     // GetIpcPtpDsParent
      UniAddr           parentUniAddr;          // GetIpcPtpDsParent
      Uint16            portNum;                // local mapping
      Uint16            portIndex;              // local mapping
} SelectedMasterEx_t;
```

4.4.5.2 SelectedMasterUni

```
typedef struct SelectedMasterUni {
      Int32             logMsgIntervalRequest;
      Int32             logMsgIntervalGrant;
      Int32             durationFieldRequest;
      Int32             durationFieldGrant;
      Uint32            txSigRutsLastMin;
      Uint32            rxSigRutsLastMin;
      Uint32            txSigGrutsLastMin;
      Uint32            rxSigGrutsLastMin;
      Int32             rxPktlogInterval;
} SelectedMasterUni_t;
```

4.4.5.3 SelectedMasterUniRep

```
typedef struct SelectedMasterUniRep {
      SelectedMasterUni_t selectedMasterUni[3];  // Announce, Sync, DelayResp
} SelectedMasterUniRep_t;
```

4.4.6 Management Port

4.4.6.1 MgmtPortStat

```
typedef struct {
      int16       ApiTotlCnt;       /* APIs counter @ 1min                    */
      int16       ApiStatus-1Cnt;   /* API STATUS=-1 error counter @ 1min     */
      int16       ApiStatus-2Cnt;   /* API STATUS=-2 busy error counter @ 1min*/
                                    /* Busy - MPU1 or MPU2 exceed 100%        */
      int16       MPU1;             /* Management port % utilization 1        */
```

```
                                    /* Recommendation: MPU1 not to exceed 100%*/
      int16       MPU2;             /* Management port % utilization 2        */
                                    /* Recommendation: MPU2 not to exceed 100%*/
      int16       ApiStatus-3Cnt;   /* API STATUS=-3 no such API error counter*/
                                    /* @1min                                  */
      int16       ApiStatus-4Cnt;   /* API STATUS=-4 device CRC error counter */
                                    /* @ 1min                                 */
      int16       ApiStatus-7Cnt;   /* API device prefix error counter @ 1min */
      int16       ApiStatus-8Cnt;   /* API with delayed service counter @ 1min*/
}MgmtPortStat;
```

4.4.7 State

4.4.7.1 GState

```
typedef struct GState {
    char state;
    char slaveStateEx;              /* in Master mode: 0                      */
    char status;                    /* device status                          */
}GState;
```

4.4.7.2 MasterCommState

```
typedef struct {
      Uint32      DelReq_state;     /*0-Down, 1-Active                        */
      Uint32      Up_elapse_time;   /*Up state elapsed time                   */
      Uint32      Dn_elapse_time;   /*Down state elapsed time                 */
      IPC_Uint48  Up_seconds;       /*Up state time                           */
      IPC_Uint48  Dn_seconds;       /*Down state time                         */
} MasterCommState;
```

4.4.7.3 SlaveManagerState

```
typedef struct {
      u_char      state;            /*0-Init,1-Free run,2-Holdover,3-Trace    */
                                    /*4-Lock                                  */
      u_char      state_extended;   /*0-Init,1-Free run,2-Holdover             */
                                    /*3:8-Trace/Lock                          */
      Uint32      FR_elapse_time;   /*Free run elapsed time                   */
      Uint32      TR_elapse_time;   /*Trace elapsed time                      */
      Uint32      LK_elapse_time;   /*Lock elapsed time                       */
      Uint32      HO_elapse_time;   /*Holdover elapsed time                   */
      Uint16      FR_seconds_H;     /*FR entry time (ptp) high                */
      Uint32      FR_seconds_L;     /*FR entry time (ptp) low                 */
      Uint16      TR_seconds_H;     /*TR entry time (ptp) high                */
      Uint32      TR_seconds_L;     /*TR entry time (ptp) low                 */
      Uint16      LK_seconds_H;     /*LK entry time (ptp) high                */
      Uint32      LK_seconds_L;     /*LK entry time (ptp) low                 */
      Uint16      HO_seconds_H;     /*HO entry time (ptp) high                */
      Uint32      HO_seconds_L;     /*HO entry time (ptp) low                 */
} SlaveManagerState;
```

4.4.7.4 ManagerState

```
typedef struct {
      u_char      state;            /*0-Init,1-Free run,2-Holdover,3-Trace    */
                                    /*4-Lock                                  */
      Uint32      FR_elapse_time;   /*Free run elapsed time                   */
      Uint32      TR_elapse_time;   /*Trace elapsed time                      */
      Uint32      LK_elapse_time;   /*Lock elapsed time                       */
      Uint32      HO_elapse_time;   /*Holdover elapsed time                   */
      Uint16      FR_seconds_H;     /*FR entry time (ptp) high                */
      Uint32      FR_seconds_L;     /*FR entry time (ptp) low                 */
      Uint16      TR_seconds_H;     /*TR entry time (ptp) high                */
      Uint32      TR_seconds_L;     /*TR entry time (ptp) low                 */
```

```
      Uint16      LK_seconds_H;     /*LK entry time (ptp) high                */
      Uint32      LK_seconds_L;     /*LK entry time (ptp) low                 */
      Uint16      HO_seconds_H;     /*HO entry time (ptp) high                */
      Uint32      HO_seconds_L;     /*HO entry time (ptp) low                 */
} ManagerState;
```

4.4.8 Statistics

4.4.8.1 Packet Statistics

```
typedef struct {
                                    /*All statistics are for past 1min        */
      Uint32      uni_rx;           /*# of Ucast pkt received                 */
      Uint32      uni_max;          /*Maximum # of Ucast received             */
      Uint32      uni_min;          /*Minimum # of Ucast received             */
      Uint32      uni_avg;          /*Average # of Ucast received             */
      Uint32      multi_rx;         /*# of Mcast pkt received                 */
      Uint32      multi_max;        /*Maximum # of Mcast received             */
      Uint32      multi_min;        /*Minimum # of Mcast received             */
      Uint32      multi_avg;        /*Average # of Mcast received             */
} pktStatistics;
```

4.4.9 Reports

4.4.9.1 SlavesServed

```
typedef struct {
      u_char      channel_valid;
      u_char      ch_id;            /*Channel ID associated with Slave        */
      Uint32      ip_address;       /*Slave IP address                        */
      MacAddr     macAddr;          /*Slave MAC address                       */
      PortIdentityA portIdentity;   /*PortIdentity                            */
      u_char      muted;            /*Slave is inactive                       */
      u_char      DelReq_connect;   /*Delay Request flow detected             */
      u_char      channel_type;     /*1-128,2-64,3-32,4-16,5-8,6-4,7-2        */
      Uint16      timeToLiveSy;     /*Time To Live                            */
      Uint16      timeToLiveDR;     /*Time To Live                            */
      Uint32      up_time_;         /*The time (PTP) the channel was up       */
      Uint32      down_time_;       /*The time (PTP) the channel was down     */
      Uint32      elapsed_up_time_; /*Up state elapsed time                   */
      Uint32      elapsed_down_time_;/*Down state elapse time                 */
      Uint16      portIndex;        /*App.for portMapMode=>1                  */
      Uint16      portNumber;       /*App.for portMapMode=>1                  */
} MasterChannelReport;

# define PAGE_SIZE = 16
typedef struct {
      MasterChannelReport           slavesServed[PAGE_SIZE];
} SlavesServed_s;

# define MASTER_MAX_CHANNELS = 64
typedef struct {
      MasterChannelReport           slavesServed[MASTER_MAX_CHANNELS];
} SlavesServedAll_s;

typedef struct {
      Uint16      AUPR;             /* Aggregated Unified Packet Rate. Group-1*/
      Uint16      MAUPR;            /* Maximal Aggregated Unified Packet Rate */
                                    /* Group-1                                */
      char        Slaves;           /* Number of channels (slaves) served     */
      char        PreScale;         /* Factor used to scale down packet rate  */
      MasterChannelReport     slavesServed_s[PAGE_SIZE];
} SlavesServed;
```

```
typedef struct {
      Uint16      AUPR;             /* Aggregated Unified Packet Rate. Group-1*/
      Uint16      MAUPR;            /* Maximal Aggregated Unified Packet Rate */
                                    /* Group-1                                */
      char        Slaves;           /* Number of channels (slaves) served     */
      char        PreScale;         /* Factor used to scale down packet rate  */
      MasterChannelReport     slavesServed_s[MASTER_MAX_CHANNELS];
} SlavesServedAll;
```

4.4.9.2 NetworkParameters

```
typedef struct {
      Uint32      time;
      int32       state;          /*0-Init,1-Free run,2-Holdover,3-Trace,4-Lock*/
      int32       commPath;         /*0-Fail, 1-Pass                          */
      int32       nativefreqOfst;  /*Native frequency Offset from Master [ppb]*/
      int32       m2s_pdvi;         /*Measured M2S PDVi in nsec (refresh@1min)*/
      int32       m2s_pdv;          /*Measured M2S PDV in nsec (refresh@10sec)*/
      int32       s2m_pdv;          /*Measured S2M PDV in nsec (refresh@10sec)*/
      int32       m2s_fpp;          /*Measured M2S FPP in 10x%                */
      int32       s2m_fpp;          /*Measured S2M FPP in 10x%                */
      int32       lockEtime;        /*Elapsed time in seconds                 */
      int32       rtd;              /*Round trip delay (1min)                 */
      int32       rtdUncertainty;   /*Round trip delay uncertainty (1min)     */
      int32       ffoffOnePps;      /*Measured FFOFF (1min)                   */
      int32       ffoffHighSpeedClk;/*Reserved                               */
} NetworkParams;
```

4.4.9.3 NetworkMinMax

```
typedef struct {
      int32       m2s_pdvi_min_;    /*Minimum PDVi in nsec                    */
      int32       m2s_pdvi_max_;    /*Maximum PDVi in nsec                    */
      int32       m2s_pdvi_ave_;    /*Average PDVi in nsec                    */

      int32       m2s_pdv_min_;     /*Minimum M2S PDV in nsec                 */
      int32       m2s_pdv_max_;     /*Maximum M2S PDV in nsec                 */
      int32       m2s_pdv_ave_;     /*Average M2S PDV in nsec                 */

      int32       s2m_pdv_min_;     /*Minimum S2M PDV in nsec                 */
      int32       s2m_pdv_max_;     /*Maximum S2M PDV in nsec                 */
      int32       s2m_pdv_ave_;     /*Average S2M PDV in nsec                 */

      int32       m2s_fpp_min_;     /*Minimum M2S FPP in 10x%                 */
      int32       m2s_fpp_max_;     /*Maximum M2S FPP in 10x%                 */
      int32       m2s_fpp_ave_;     /*Average M2S FPP in 10x%                 */

      int32       s2m_fpp_min_;     /*Minimum S2M FPP in 10x%                 */
      int32       s2m_fpp_max_;     /*Maximum S2M FPP in 10x%                 */
      int32       s2m_fpp_ave_;     /*Average S2M FPP in 10x%                 */

      int32       td_min_;          /*Minimum RTD in nsec                     */
      int32       td_max_;          /*Maximum RTD in nsec                     */
      int32       td_ave_;          /*Average RTD in nsec                     */
      int32       tu_min_;          /*Minimum RTD uncertainty in nsec         */
      int32       tu_max_;          /*Maximum RTD uncertainty in nsec         */
      int32       tu_ave_;          /*Average RTD uncertainty in nsec         */
      int32       ffoff_1pps_max_;  /*Maximal measured FFOFF                  */
      int32       ffoff_10M_max_;   /*Reserved                                */
} NetworkMinMax;
```

4.4.9.4 PacketCntSlave

```
typedef struct {
      Uint32       RxAn;
      Uint32       RxAnDrop;
      Uint32       RxSync;
      Uint32       RxSyncMiss;
      Uint32       RxSyncMissOrd;
      Uint32       RxSyncDrop;
      Uint32       RxFU;
      Uint32       RxFUMiss;
      Uint32       RxFUDrop;
      Uint32       RxDResp;
      Uint32       RxDRespMiss;
      Uint32       RxDRespMissOrd;
      Uint32       RxDRespDrop;
      Uint32       RxDRespReqSrcPortIdDrop;
      Uint32       TxDReq;
} PacketCntSlave;
```

4.4.9.5 PacketCntMaster

```
typedef struct {
      Uint32       TxAn;
      Uint32       TxSync;
      Uint32       TxFU;
      Uint32       RxDReq;
      Uint32       ReDReqDrop1;
      Uint32       RxDReqDrop2;
      Uint32       RxDReqDrop3;
      Uint32       TxDResp;
} PacketCntMaster;
```

4.4.9.6 PacketCntSignaling

```
typedef struct {
      Uint32       txSigRuts;
      Uint32       rxSigRuts;
      Uint32       txSigGruts;
      Uint32       rxSigGruts;
      Uint32       txSigDgruts;
      Uint32       rxSigDgruts;
      Uint32       txSigCuts;
      Uint32       rxSigCuts;
      Uint32       txSigAcuts;
      Uint32       rxSigAcuts;
} PacketCntSignaling_t;

typedef struct PacketCntSignaling {
      PacketCntSignaling_t packetCntSignaling[3];        // Ann, Sync, DelayResp
} PacketCntSignaling_s;

typedef struct PacketCntSignalingLastMin {
      PacketCntSignaling_t packetCntSignalingLastMin[3]; // Ann, Sync, DelayResp
} PacketCntSignalingLastMin_s;
```

4.4.9.7 PacketCntPort

```
typedef struct PacketCntPort {
      Uint32       TxAn;
      Uint32       RxAn;
      Uint32       RxAnStateDrop;
      Uint32       TxSync;
      Uint32       RxSync;
      Uint32       RxSyncStateDrop;
      Uint32       TxFU;
      Uint32       RxFU;
      Uint32       RxFUStateDrop;
      Uint32       TxDReq;
```

```
        Uint32        RxDReq;
        Uint32        RxDReqStateDrop;
        Uint32        TxDResp;
        Uint32        RxDResp;
        Uint32        RxDRespStateDrop;
        Uint32        TxSig;
        Uint32        RxSig;
        Uint32        RxSigStateDrop;
        Uint32        TxMgmt;
        Uint32        RxMgmt;
        Uint32        RxMgmtStateDrop;
} PacketCntPort_t;

# define PAGE_SIZE = 16
typedef struct PacketCntPort {
        PacketCntPort_t          packetCntPort[PAGE_SIZE];
} PacketCntPort_s;

# define PORT_INDEX_MAX_ = 64
typedef struct PacketCntPortAll {
        PacketCntPort_t          packetCntPort[PORT_INDEX_MAX];
} PacketCntPortAll_s;
```

4.4.9.8 PacketCntOther

```
typedef struct PacketCntOther {
        Uint32        rxNonMapped;              // all packet types
        Uint32        txStateDrop;              // all packet types
        Uint32        rxSlaveSyncPortNumDrop;   // S
        Uint32        rxSlaveFuPortNumDrop;
        Uint32        rxSlaveDelayReqPortNumDrop;
} PacketCntOther_t;
```

4.4.9.9 PDVHist

```
typedef struct {
        /*                                M2S PDV Histogram                        */
        int32 M2SPdvHistNsMax;
        int32 M2SPdvHistFloorCountN;  // Count - Bin1 to BinN (Bin6, <150us)
                                      // Number of packets in the master to
                                      // slave path (Sync/FU) in Bin1-Bin6,
                                      // PDV < 150us
        int32 M2SPdvHistFloorCountM;  // Count - Bin1 to BinM (Bin4, <50us)
                                      // Number of packets in the master to
                                      // slave path (Sync/FU) in Bin1-Bin4,
                                      // PDV < 50us
        int32 M2SPdvHistValidPktCount;// Bin1 to Bin25
        int32 M2SPdvHistTotalPktCount;// Bin1 to Bin25, Bin101, Bin102
        int32 M2SPdvHistFloorPercentN;// 10xPercent - Bin1 to BinN (Bin6, <150us)
        int32 M2SPdvHistFloorPercentM;// 10xPercent - Bin1 to BinM (Bin4, <50us)
        int32 M2SPdvHistValidPercent; // 10xPercent - Bin1 to Bin25
        int32 M2SPdvHistEtSec;
        int32 M2SPdvHistNsMin;        // Reserved
        int32 M2SPdvHistBLH;          // Reserved G012 sec
        int32 M2SPdvHistBLL;          // Reserved 10xG012 nsec

        /* Each bin contains 10xPercent, Count, NCount                            */
                                      /*msec  : msec                              */
        int32 M2SPdvHistBin1[3];      /*      : 0.0005                           */
        int32 M2SPdvHistBin2[3];      /*0.0005: 0.005                            */
        int32 M2SPdvHistBin3[3];      /*0.005 : 0.010                            */
        int32 M2SPdvHistBin4[3];      /*0.01  : 0.05                             */
        int32 M2SPdvHistBin5[3];      /*0.05  : 0.10                             */
        int32 M2SPdvHistBin6[3];      /*0.10  : 0.15                             */
        int32 M2SPdvHistBin7[3];      /*0.15  : 0.2                              */
```

```
int32 M2SPdvHistBin8[3];       /*0.2   : 0.3                              */
int32 M2SPdvHistBin9[3];       /*0.3   : 0.4                              */
int32 M2SPdvHistBin10[3];      /*0.4   : 0.5                              */
int32 M2SPdvHistBin11[3];      /*0.5   : 0.6                              */
int32 M2SPdvHistBin12[3];      /*0.6   : 0.7                              */
int32 M2SPdvHistBin13[3];      /*0.7   : 0.8                              */
int32 M2SPdvHistBin14[3];      /*0.8   : 0.9                              */
int32 M2SPdvHistBin15[3];      /*0.9   : 1.0                              */
int32 M2SPdvHistBin16[3];      /*1.0   : 2.0                              */
int32 M2SPdvHistBin17[3];      /*2.0   : 3.0                              */
int32 M2SPdvHistBin18[3];      /*3.0   : 4.0                              */
int32 M2SPdvHistBin19[3];      /*4.0   : 5.0                              */
int32 M2SPdvHistBin20[3];      /*5.0   : 6.0                              */
int32 M2SPdvHistBin21[3];      /*6.0   : 7.0                              */
int32 M2SPdvHistBin22[3];      /*7.0   : 8.0                              */
int32 M2SPdvHistBin23[3];      /*8.0   : 9.0                              */
int32 M2SPdvHistBin24[3];      /*9.0   :10.0                              */
int32 M2SPdvHistBin25[3];      /*10.0  :100                               */
int32 M2SPdvHistBin101[3];     /*Reserved - Not Valid                     */
int32 M2SPdvHistBin102[3];     /*Reserved - Not Valid                     */

/*                                S2M PDV Histogram                       */
int32 S2MPdvHistNsMax;
int32 S2MPdvHistFloorCountN;   // Count - Bin1 to BinN (Bin6, <150us)
                               // Number of packets in the slave to
                               // master path (DelayReq) in Bin1-Bin6,
                               // PDV < 150us
int32 S2MPdvHistFloorCountM;   // Count - Bin1 to BinM (Bin4, <50us)
                               // Number of packets in the slave to
                               // master path (DelayReq) in Bin1-Bin4,
                               // PDV < 50us
int32 S2MPdvHistValidPktCount;// Bin1 to Bin25
int32 S2MPdvHistTotalPktCount;// Bin1 to Bin25, Bin101, Bin102
int32 S2MPdvHistFloorPercentN;// 10xPercent - Bin1 to BinN (Bin6, <150us)
int32 S2MPdvHistFloorPercentM;// 10xPercent - Bin1 to BinM (Bin4, <50us)
int32 S2MPdvHistValidPercent; // 10xPercent - Bin1 to Bin25
int32 S2MPdvHistEtSec;
int32 S2MPdvHistNsMin;         // Reserved
int32 S2MPdvHistBLH;           // Reserved G034 sec
int32 S2MPdvHistBLL;           // Reserved 10xG034 nsec

/* Each bin contains 10xPercent, Count, NCount                            */
                               /*msec  : msec                             */
int32 S2MPdvHistBin1[3];       /*      : 0.0005                           */
int32 S2MPdvHistBin2[3];       /*0.0005: 0.005                            */
int32 S2MPdvHistBin3[3];       /*0.005 : 0.010                            */
int32 S2MPdvHistBin4[3];       /*0.01  : 0.05                             */
int32 S2MPdvHistBin5[3];       /*0.05  : 0.10                             */
int32 S2MPdvHistBin6[3];       /*0.10  : 0.15                             */
int32 S2MPdvHistBin7[3];       /*0.15  : 0.2                              */
int32 S2MPdvHistBin8[3];       /*0.2   : 0.3                              */
int32 S2MPdvHistBin9[3];       /*0.3   : 0.4                              */
int32 S2MPdvHistBin10[3];      /*0.4   : 0.5                              */
int32 S2MPdvHistBin11[3];      /*0.5   : 0.6                              */
int32 S2MPdvHistBin12[3];      /*0.6   : 0.7                              */
int32 S2MPdvHistBin13[3];      /*0.7   : 0.8                              */
int32 S2MPdvHistBin14[3];      /*0.8   : 0.9                              */
int32 S2MPdvHistBin15[3];      /*0.9   : 1.0                              */
int32 S2MPdvHistBin16[3];      /*1.0   : 2.0                              */
int32 S2MPdvHistBin17[3];      /*2.0   : 3.0                              */
int32 S2MPdvHistBin18[3];      /*3.0   : 4.0                              */
int32 S2MPdvHistBin19[3];      /*4.0   : 5.0                              */
int32 S2MPdvHistBin20[3];      /*5.0   : 6.0                              */
```

```
      int32 S2MPdvHistBin21[3];     /*6.0   : 7.0                              */
      int32 S2MPdvHistBin22[3];     /*7.0   : 8.0                              */
      int32 S2MPdvHistBin23[3];     /*8.0   : 9.0                              */
      int32 S2MPdvHistBin24[3];     /*9.0   :10.0                              */
      int32 S2MPdvHistBin25[3];     /*10.0  :100                               */
      int32 S2MPdvHistBin101[3];    /*Reserved - Not Valid                     */
      int32 S2MPdvHistBin102[3];    /*Reserved - Not Valid                     */
} PdvHist;
```

4.4.9.10 FppBuffer

```
# define FPP_MAX_SIZE = 100   // for minimal memory FPP_MAX_SIZE = 20
PdvHistShort      fppBuf[FPP_MAX_SIZE];
```

4.4.9.11 FppPar

```
typedef struct {
      int32 FppMode=1;      // 0-Disable, 1-Enable
      int32 FppDeltaM=50;   // usec, M <= N
      int32 FppDeltaN=150;  // usec
      int32 FppWinSec=200;  // sec
      int32 FppStepSec=10;  // sec. must be set to 10.
      int32 FppRep=1;       // 0-Disable
                            // 1-Report msg 0050,1350
                            // 2-Report msg 0050,1350 & 0469 M2S/S2M every step
                            // 3-Report msg 0050,1350 & 0469 every step
      int32 FppBistTh=10;   // 10 = 1%, 1000 = 100%
      int32 PdvBistTh=10;   // in ms. 10 = 10ms
} FppPar;
```

4.4.9.12 FppRep

```
typedef struct {
      int32 WinTime;         // Window Time fill level in sec
      // M2S
      int32 M2SCntM;         // Sum(FppBuf[n].M2SPdvHistFloorCountM) over window
                             // Number of packets in the master to slave path
                             // (Sync/FU) in Bin1-Bin6, PDV < 150us over
                             // sliding window of 200sec
      int32 M2SCntN;         // Sum(FppBuf[n].M2SPdvHistFloorCountN) over window
                             // Number of packets in the master to slave path
                             // (Sync/FU) in Bin1-Bin4, PDV < 50us over
                             // sliding window of 200sec
      Int32 M2SVPkt;         // Sum(FppBuf[n].M2SPdvHistValidPkts) over window
      Int32 M2STPkt;         // Sum(FppBuf[n].M2SPdvHistTotalPkts) over window
      int32 M2SUC;           // p2p(FppBuf[n].M2SPdvHistBLL/H) over window
                             // IF M2SUC > 10000 THEN  M2SUC = 10000
      int32 M2SPdvHistNsMax;// Max(FppBuf[n].M2SPdvHistNsMax) over window
      int32 M2SPercentageM; // FppRep.M2SCntM * 1000 / FppRep.M2SVPkt
                             // IF M2SVPkt = 0   THEN  M2SPercentageM = -1
      int32 M2SPercentageN; // FppRep.M2SCntN * 1000 / FppRep.M2SVPkt
                             // IF M2SVPkt = 0   THEN  M2SPercentageN = -1
      int32 M2SPercentageV; // FppRep.M2SVPkt * 1000 / FppRep.M2STPkt
                             // IF M2STPkt = 0   THEN  M2SPercentageV = -1
      // S2M
      int32 S2MCntM;         // Sum(FppBuf[n].S2MPdvHistFloorCountM) over window
                             // Number of packets in the slave to master path
                             // (DelayReq) in Bin1-Bin6, PDV < 150us over
                             // sliding window of 200sec
      int32 S2MCntN;         // Sum(FppBuf[n].S2MPdvHistFloorCountN) over window
                             // Number of packets in the slave to master path
                             // (DelayReq) in Bin1-Bin4, PDV < 50us over
                             // sliding window of 200sec
      Int32 S2MVPkt;         // Sum(FppBuf[n].S2MPdvHistValidPkts) over window
```

```
      Int32 S2MTPkt;         // Sum(FppBuf[n].S2MPdvHistTotalPkts) over window
      int32 S2MUC;           // p2p(FppBuf[n].S2MPdvHistBLL/H) over window
                             // IF S2MUC > 10000 THEN  S2MUC = 10000
      int32 S2MPdvHistNsMax;// Max(FppBuf[n].S2MPdvHistNsMax) over window
      int32 S2MPercentageM; // FppRep.S2MCntM * 1000 / FppRep.S2MVPkt
                             // IF S2MVPkt = 0   THEN  S2MPercentageM = -1
      int32 S2MPercentageN; // FppRep.S2MCntN * 1000 / FppRep.S2MVPkt
                             // IF S2MVPkt = 0   THEN  S2MPercentageN = -1
      int32 S2MPercentageV; // FppRep.S2MVPkt * 1000 / FppRep.S2MTPkt
                             // IF S2MTPkt = 0   THEN  S2MPercentageV = -1
} FppRep;
```

4.4.9.13 Fpp

```
typedef struct {
      int16 M2SPercentageN; // FppRep.S2MPercentageN (10x%)
      int32 M2SPdvHistNsMax;// FppRep.M2SPdvHistNsMax
      int16 S2MPercentageN; // FppRep.S2MPercentageN (10x%)
      int32 S2MPdvHistNsMax;// FppRep.S2MPdvHistNsMax
} Fpp;
```

4.4.9.14 Frequency Estimation

```
typedef struct FreqEstRep {
      FRQi[60];
} FreqEstRep;
```

4.4.9.15 Log Buffer Status

```
typedef struct TxSysLogBufferStatus {
      int16 logUtilizationChar;        /* buffer size 20000 char              */
      int16 logUtilizationPercentage;  /* 100% = 20000 char                   */
      int16 logOverflow;               /* 1 indicates overflow. Clear on read */
      int16 apiCnt;                    /* number of GetIpcAdminLog call.      */
                                       /* Clear on read                       */
} TxSysLogBufferStatus;
```

4.4.9.16 portMembers

```
#define PORT_INDEX_MAX 64                  // for 64 channels
typedef struct PortMembers{
      Uint16 portMembers[PORT_INDEX_MAX]; // init to 0.
} PortMembers_t;
```

4.4.10 BIST

4.4.10.1 SlaveBistInit

```
typedef struct {
      TBistTestResult_t hwRead;           /*Reading pre-determined data from  */
                                          /*a pre-determined register         */
      TBistTestResult_t hwReadWrite1;     /*Writing data to a register and    */
                                          /*reading back                      */
      TBistTestResult_t hwReadWrite2;     /*Writing data to a register and    */
                                          /*reading back                      */
      TBistTestResult_t Pll1Lock;         /*PLL-1 is locked                   */
      TBistTestResult_t Pll2Lock;         /*PLL-2 is locked                   */
      TBistTestResult_t Pll3Lock;         /*PLL-3 is locked                   */
      TBistTestResult_t reserved1;        /*Reserved                          */
      TBistTestResult_t reserved2;        /*Reserved                          */
      TBistTestResult_t reserved3;        /*Reserved                          */
      TBistTestResult_t clockSyncInit;    /*Clock synchronization init        */
      TBistTestResult_t reserved4;        /*Reserved                          */
      TBistTestResult_t reserved5;        /*Reserved                          */
      TBistTestResult_t initPassed;       /*Init BIST passed                  */
```

```
} SlaveBistInit;

// IPC9600
typedef struct {
      TBistTestResult_t hwRead;           /*Reading pre-determined data from  */
                                          /*a pre-determined register         */
      TBistTestResult_t hwReadWrite1;     /*Writing data to a register and    */
                                          /*reading back                      */
      TBistTestResult_t hwReadWrite2;     /*Writing data to a register and    */
                                          /*reading back                      */
      TBistTestResult_t Pll1Lock;         /*PLL-1 is locked                   */
      TBistTestResult_t Pll2Lock;         /*PLL-2 is locked                   */
      TBistTestResult_t Pll3Lock;         /*PLL-3 is locked                   */
      TBistTestResult_t Pll4Lock;         /*PLL-4 is locked                   */
      TBistTestResult_t Pll5Lock;         /*PLL-5 is locked                   */
      TBistTestResult_t Pll5Normal;       /*PLL-5 is in normal operation      */
                                          /*PLL-5 was not reset               */
      TBistTestResult_t clockSyncInit;    /*Clock synchronization init        */
      TBistTestResult_t edDetect;         /*External device detect            */
      TBistTestResult_t edInit;           /*External device initialization    */
      TBistTestResult_t initPassed;       /*Init BIST passed                  */
} SlaveBistInit;
```

4.4.10.2 MasterBistInit

```
typedef struct {
      TBistTestResult_t hwRead;           /*Reading pre-determined data from  */
                                          /*a pre-determined register         */
      TBistTestResult_t hwReadWrite1;     /*Writing data to a register and    */
                                          /*reading back                      */
      TBistTestResult_t hwReadWrite2;     /*Writing data to a register and    */
                                          /*reading back                      */
      TBistTestResult_t Pll1Lock;         /*PLL-1 is locked (N/A for BC)      */
      TBistTestResult_t Pll2Lock;         /*PLL-2 is locked (N/A for BC)      */
      TBistTestResult_t Pll3Lock;         /*PLL-3 is locked (N/A for BC)      */
      TBistTestResult_t reserved1;        /*Reserved                          */
      TBistTestResult_t reserved2;        /*Reserved                          */
      TBistTestResult_t reserved3;        /*Reserved                          */
      TBistTestResult_t clockSyncInit;    /*Clock synchronization init        */
      TBistTestResult_t initPassed;       /*Init BIST passed                  */
} MasterBistInit;

// IPC9600
typedef struct {
      TBistTestResult_t hwRead;           /*Reading pre-determined data from  */
                                          /*a pre-determined register         */
      TBistTestResult_t hwReadWrite1;     /*Writing data to a register and    */
                                          /*reading back                      */
      TBistTestResult_t hwReadWrite2;     /*Writing data to a register and    */
                                          /*reading back                      */
      TBistTestResult_t Pll1Lock;         /*PLL-1 is locked (N/A for BC)      */
      TBistTestResult_t Pll2Lock;         /*PLL-2 is locked (N/A for BC)      */
      TBistTestResult_t Pll3Lock;         /*PLL-3 is locked (N/A for BC)      */
      TBistTestResult_t Pll4Lock;         /*PLL-4 is locked (N/A for BC)      */
      TBistTestResult_t Pll5Lock;         /*PLL-5 is locked (N/A for BC)      */
      TBistTestResult_t Pll5Normal;       /*PLL-5 is in normal operation      */
                                          /*PLL-5 was not reset (N/A for BC)  */
      TBistTestResult_t clockSyncInit;    /*Clock synchronization init        */
      TBistTestResult_t edDetect;         /*External device detect(N/A for BC)*/
      TBistTestResult_t edInit;           /*External device initalization     */
                                          /*(N/A for BC)                      */
      TBistTestResult_t initPassed;       /*Init BIST passed                  */
} MasterBistInit;
```

4.4.10.3 SlaveBistShowtime

```
typedef struct {
      TBistTestResult_t localClk;         /*Local Clock detected              */
      TBistTestResult_t Reserved;         /*Reserved                          */
      TBistTestResult_t rxAnnounce;       /*Valid Announce packets received   */
      TBistTestResult_t commPath;         /*Communication path status         */
      TBistTestResult_t ipcState;         /*State is active                   */
      TBistTestResult_t oscFreqOfst;      /*Local oscillator frequency offset */
                                          /*is within limits                  */
      TBistTestResult_t pdvOk;            /*PDV is within limits              */
      TBistTestResult_t ReservedA;        /*Reserved                          */
      TBistTestResult_t ReservedB;        /*Reserved                          */
      TBistTestResult_t ReservedC;        /*Reserved                          */
      TBistTestResult_t ReservedD;        /*Reserved                          */
      TBistTestResult_t parentMasterLk;   /*Selected (Parent) master report   */
                                          /*to be in Lock state               */
      TBistTestResult_t hwRxPkts;         /*Hardware is detecting and         */
                                          /*classifying 1588v2 packets        */
      TBistTestResult_t Reserved1;        /*Reserved                          */
      TBistTestResult_t Reserved2;        /*Reserved                          */
      TBistTestResult_t Reserved3;        /*Reserved                          */
      TBistTestResult_t gState;           /*Global state                      */
      TBistTestResult_t showtimePassed;   /*Showtime BIST passed              */
} SlaveBistShowTime;

// IPC9600
typedef struct {
      TBistTestResult_t localClk;         /*Local Clock detected              */
      TBistTestResult_t Reserved;         /*Reserved                          */
      TBistTestResult_t rxAnnounce;       /*Valid Announce packets received   */
      TBistTestResult_t commPath;         /*Communication path status         */
      TBistTestResult_t ipcState;         /*State is active                   */
      TBistTestResult_t oscFreqOfst;      /*Local oscillator frequency offset */
                                          /*is within limits                  */
      TBistTestResult_t pdvOk;            /*PDV is within limits              */
      TBistTestResult_t fppM2S;           /*FPP M2S is within limits          */
      TBistTestResult_t fppS2M;           /*FPP S2M is within limits          */
      TBistTestResult_t pdvM2S;           /*PDV M2S is within limits          */
      TBistTestResult_t pdvS2M;           /*PDV S2M is within limits          */
      TBistTestResult_t parentMasterLk;   /*Selected (Parent) master report   */
                                          /*to be in Lock state               */
      TBistTestResult_t hwRxPkts;         /*Hardware is detecting and         */
                                          /*classifying 1588v2 packets        */
      TBistTestResult_t edClkInValid;     /*External device clock in valid    */
      TBistTestResult_t edCtrl;           /*External device control           */
      TBistTestResult_t edTrk;            /*External device track             */
      TBistTestResult_t gState;           /*Global state                      */
      TBistTestResult_t showtimePassed;   /*Showtime BIST passed              */
} SlaveBistShowTime;
```

4.4.10.4 MasterBistShowtime

```
# define PAGE_SIZE = 16
typedef struct {
      TBistTestResult   txSync;           /*Sync packets are transmitted      */
      TBistTestResult   localClk          /*Local Clock detected              */
      TBistTestResult   Reserved;         /*Reserved                          */
      TBistTestResult   Reserved;         /*Reserved                          */
      TBistTestResult   ppsLock;          /*Locked to 1PPS (N/A for BC)       */
      TBistTestResult   timeStamp;        /*Hardware timestamp is active      */
      TBistTestResult_t edTrk;            /*External device track(N/A for BC) */
      TBistTestResult_t gState;           /*Global state (N/A for BC)         */
      TBistTestResult_t txSyncCh[PAGE_SIZE];/*Channel Tx Sync packets         */
```

```
      TBistTestResult_t keepAlive[PAGE_SIZE]; /*Ch keep alive status          */
      TBistTestResult_t chStat[PAGE_SIZE];/*Channel commpath                 */
      TBistTestResult_t showtimePassed;   /*Showtime BIST passed              */
} MasterBistShowTime;
```

4.4.10.5 BistNativePll

```
// IPC9600
typedef struct {
      TBistTestResult_t Pll1NativeLock;   /*PLL-1 native lock                 */
                                          /*Reg 0x30 bit 8 (0x30[8])          */
      TBistTestResult_t Pll2NativeLock;   /*PLL-2 native lock                 */
                                          /*Reg 0x30 bit 11 (0x30[11])        */
      TBistTestResult_t Pll3NativeLock;   /*PLL-3 native lock                 */
      TBistTestResult_t Pll4NativeLock;   /*PLL-4 native lock                 */
                                          /*Reg 0x30 bit 9 (0x30[9])
      TBistTestResult_t Pll4NativeNormal; /*PLL-4 native lock normal operation*/
                                          /*PLL-4 was not lock loss           */
                                          /*Clear on write 1                  */
                                          /*NOT(Reg 0x36 bit 7 (0x36[7]))     */
      TBistTestResult_t Pll5NativeLock;   /*PLL-5 native lock                 */
                                          /*NOT (Reg 0x30 bit 10 (0x30[10]))  */
      TBistTestResult_t Pll5NativeNormal; /*PLL-5 native lock normal operation*/
                                          /*PLL-5 was not reset               */
                                          /*Clear on write 1                  */
                                          /*NOT(Reg 0x36 bit 15 (0x36[15]))   */
      TBistTestResult_t PllPassed;        /*PLL BIST passed                   */
} BistNativePll;
```

4.4.10.6 BistPll

```
// IPC9600
typedef struct {
      TBistTestResult_t Pll1Lock;   /*PLL-1 lock                 */
      TBistTestResult_t Pll2Lock;   /*PLL-2 lock                 */
      TBistTestResult_t Pll3Lock;   /*PLL-3 lock                 */
      TBistTestResult_t Pll4Lock;   /*PLL-4 lock                 */
      TBistTestResult_t Pll4Normal; /*PLL-4 lock normal operation */
      TBistTestResult_t Pll5Lock;   /*PLL-5 lock                 */
      TBistTestResult_t Pll5Normal; /*PLL-5 lock normal operation */
      TBistTestResult_t PllPassed;  /*PLL BIST passed            */
} BistPll;
```

4.4.11 Packet Dump Mode

```
typedef struct {
      int32 PktDumpMode.RxTx;             /* Tx or Rx                       */
      int32 PktDumpMode.MessageTypeEn;    /* Filteration per packet type    */
      int32 PktDumpMode.MessageType;      /* Packet type                    */
      int32 PktDumpMode.LocalTSDump;      /* Local TS dump                  */
      int32 PktDumpMode.Number;           /* Number of consecutive packets  */
} PktDumpMode
```

4.4.12 Unicast

4.4.12.1 PortAddress

```
struct SIeee1588Prf_PortAddress {
      Uint16 networkProtocol;       /* 1-UDP/IPv4, 3-IEEE 802.3            */
      Uint16 addressLength;         /* 4-UDP/IPv4, 6-IEEE 802.3            */
      UniAddr addressField;         /* Uniform Address                     */
      char   hwAddr[6];             /* MAC address                         */
      PortIdentityA portIdentity;   /*PortIdentity                         */
} SIeee1588PrfPortAddress_t;
```

4.4.12.2 PortAddressQueryTable

```
#define NP_MAX_PAQT_SIZE      16

Struct SIeee1588Prf_PortAddressQueryTable {
      Uint16 maxTableSize;          /*NP_MAX_PAQT_SIZE                    */
      Uint16 actualTableSize;       /*Default = 0                         */
      SIeee1588PrfPortAddress_t portAddress[NP_MAX_PAQT_SIZE];
} SIeee1588PrfPortAddressQueryTable_t;
```

4.4.12.3 AcceptableMaster

```
typedef Struct SIeee1588Prf_AcceptableMaster {
      SIeee1588PrfPortAddress_t portAddress;
      Uint16      alternatePriority1;     /*Replacing Rx Announce Priority1   */
                                          /*if value <> 0, Default = 0        */
} SIeee1588PrfAcceptableMaster_t;
```

4.4.12.4 AcceptableMasterTable

```
#define NP_MAX_AMT_SIZE       8

typedef struct SIeee1588Prf_AcceptableMasterTable {
      Uint16      maxTableSize;     /*Max = 8                             */
      Uint16      actualTableSize;  /*Default =0                          */
      SIeee1588PrfAcceptableMaster_t acceptableMaster[NP_MAX_AMT_SIZE];
} SIeee1588PrfAcceptableMasterTable_t;
```

4.4.12.5 AcceptableMasterExtended

```
typedef struct SIeee1588Prf_AcceptableMasterExtended {
      SIeee1588Prf_PortAddress      portAddress;
      Uint16      alternatePriority1;     /* Replacing Announce              */
      int32       ptsfMode;               /* Indicates the PTSF mode:        */
                                          /* 0-OFF(Pass), 1-ON (Fail), 3-AUTO */
      int32       ptsfLossTotal;          /* Loss of any PTSF                */
      int32       ptsfLossSync;           /* Loss of sync/delay_resp         */
      int32       ptsfLossAnnounce;       /* Loss of Announce                */
      int32       ptsfWaitToRestore;      /* Wait to restore                 */
      Uint32      rxAnnCnt;               /* Received Announce packets       */
      Uint32      rxAnnCntOld;            /* Not presented in Terminal mode  */
      Uint32      lossAnnSecCnt;          /* Not presented in Terminal mode  */
      Uint32      rxSyncCnt;              /* Received Sync packets           */
      Uint32      rxDrspCnt;              /* Received Delay Resp packets     */
      Uint32      masterSel;              /* Selected: masterSel=1           */
} SIeee1588PrfAcceptableMasterExtended_t;
```

4.4.12.6 AcceptableMasterTableExtended

```
typedef struct SIeee1588Prf_AcceptableMasterTableExtended {
      Uint16      maxTableSize;           /* Max = 8              */
      Uint16      actualTableSize;        /* Default =0           */
      SIeee1588PrfAcceptableMasterExtended_t
            acceptableMasterExtended[NP_MAX_AMT_SIZE];
} SIeee1588PrfAcceptableMasterTableExtended;
```

4.4.12.7 UnicastPtsfConfig

```
typedef struct SIeee1588Prf_UnicastPtsfConfig {
    Uint32              ptsfSyncTh;
    Uint32              ptsfAnnounceTh;
    Uint32              ptsfUnusableTh;
    Uint32              ptsfUnusablePdvTh;
    Uint32              ptsfWaitToRestoreTh;
} SIeee1588PrfUnicastPtsfConfig_t;
```

4.4.12.8 RutsEn

```
typedef struct {
      int32 syncPkt;          /*Slave Sync RUTS enable                    */
      int32 delayRespPkt;     /*Slave Delay Resp RUTS enable              */
      int32 announcePkt;      /*Slave Announce RUTS enable                */
} RutsEn;
```

4.4.12.9 RutsDuration

```
typedef struct {
      int32 rutsDurationAn;
      int32 rutsDurationSy;
      int32 rutsDurationDR;
} RutsDuration;
```

4.4.12.10 AnnounceTableEntry

```
typedef struct SIeee1588Prf_AnnounceTableEntry {
      Uint32      addressField;                /*IPv4 Address              */
      PortIdentityA portIdentity;              /*PortIdentity              */
      Int16       logPacketPeriod;             /*log2 of packet period     */
      Int16       reqlogPacketPeriod;          /*Req. log2 of packet period*/
      Uint16      timeToLive;                  /*10-1000 sec               */
      Uint16      seqId;                       /*Last Announce Sequence ID */
      Uint16      portIndex;                   /*App.for portMapMode=>1    */
      Uint16      portNum;                     /*App.for portMapMode=>1    */
      Uint16      active;                      /*App.for portMapMode=>1    */
} AnnounceTableEntry;
```

4.4.12.11 AnnounceTable

```
# define MASTER_MAX_CHANNELS = 64
typedef struct {
      Uint16      maxTableSize;     /* 64                                  */
      Uint16      actualTableSize;  /* Default = 0                         */
      Uint16      AUPR;             /* Aggregated Unified Packet Rate. Group-2*/
      Uint16      MAUPR;            /* Maximal Aggregated Unified Packet Rate */
                                    /* Group-2                             */
      AnnounceTableEntry      AnnounceTableEntry[NP_MAX_GRAT_SIZE];
} AnnounceTable;
```

4.4.12.12 ForeignMaster

```
typedef struct ForeignMaster {
      UniAddr           sourceUniAddr;
      PortIdentity_t    parentPortIdentity;
      ClockIdentity_t   grandmasterIdentiy;
      u_char            domain;
      int32             expirationCounter; // TTL
      u_char            priority1;
      u_char            priority2;
      Uint16            stepsRemoved;
      ClockQuality_t    clockQuality;
      u_char            localPriority; // Applicable for ptpBmcMode=10
      u_char            notSlave;      // Applicable for ptpBmcMode=10
      Uint16            portIndex;     // Applicable for portMapMode=>1
      Uint16            portNumber;    // Applicable for portMapMode=>1
} ForeignMaster_t;
```

4.4.12.13 ForeignMasterMainTable

```
#define MAIN_MASTERS_TABLE_SIZE 16

typedef struct ForeignMasterMainTable {
      Uint16            maxTableSize;     /* MAIN_MASTERS_TABLE_SIZE       */
```

```
      Uint16            actualTableSize;  /* Default = 0                    */
      ForeignMaster_t   foreignMaster[MAIN_MASTERS_TABLE_SIZE];
} ForeignMasterMainTable_t;
```

4.4.13 Port Mapping

4.4.13.1 AcceptableAddr

```
typedef Struct AcceptableAddr {
      UniAddr           addressField;     /* Uniform Address                */
      Uint16            portIndex;
      Uint16            portNumber;
      u_char            portState;
} AcceptableAddr_t;
```

4.4.13.2 AcceptableTable

```
#define MAT_PAGE        16
typedef struct AcceptableTable {
      Uint16            maxTableSize;           /*Max = 64                 */
      Uint16            actualTableSize;        /*Default =0               */
      AcceptableAddr_t  acceptableTable[MAT_PAGE];
} AcceptableTable_t;
```

4.4.13.3 AcceptableTableAll

```
#define MAT_SIZE        64
typedef struct AcceptableTableAll {
      Uint16            maxTableSize;           /*Max = 64                 */
      Uint16            actualTableSize;        /*Default =0               */
      AcceptableAddr_t  acceptableTable[MTP2_SIZE];
} AcceptableTableAll_t;
```

4.4.14 APTS

4.4.14.1 VirtualPtpPort

```
typedef struct virtualPtpPort {
      u_char reserved1;
      u_char clockClassMode;
      u_char clockClassConf;
      u_char clockClass;
      u_char clockAccuracy;
      Uint16 offsetScaledLogVar;
      Uint16 portNum;
      u_char localPriority;   //Applicable for ptpBmcMode=10
} virtualPtpPort;
```

4.4.14.2 AptsPar

```
typedef struct aptsPar {
      int32 getParentDsMode;
      int32 announceSrcParentDsMode;
      int32 stepsRemovedControlledMode;
      int32 stateMode
} aptsPar;
```

4.4.14.3 AscInfo

```
typedef struct {
      int32 cnt     = 0; // number of time-updates by AsymComp.
      int32 lastCorr= 0; // last asymmetry compensation corr. [ns]
      int32 aggCorr = 0; // aggregated asymmetry compensation corr [ns]
      int32 totalCorr=0; // Total asymmetry compensation corr. [ns]
} ascInfo;
```

4.4.15 External Device

Applicable for IPC9600 only.

4.4.15.1 EDStatus

```
// External Device Status.
typedef struct {
      int32 EDStatus.G_ST;// Global Status
                          // same as slave status
                          // 0 – Init / No Signal
                          // 1 – Free run
                          // 2 – Holdover
                          // 3 – Trace
                          // 4 – Lock
      int32 EDStatus.ED_ST;// External Device Status
                          // 0 – Init
                          // 1 – Reserved (extDevMode=41/45)
                          //     Free run (extDevMode=42/43/44/47)
                          // 2 – No signal / No valid signal (extDevMode=41/45)
                          //     Holdover (extDevMode=42/43/44/47)
                          // 3 – Trace
                          // 4 – Lock
      int32 EDStatus.ED_STB;// External Device B Status
                          // Copy from AdiState
      int32 EDStatus.ST;  // state from GetIpcPtpState
                          // Freq after limiter.
                          // Applicable for BC and Slave with extDevMode 41,45
      int32 EDStatus.F_Mn;// minimal Freq command of the last minute [ppb,10^-9]
      int32 EDStatus.F_Mx;// maximal Freq command of the last minute [ppb,10^-9]
      int32 EDStatus.F_Av;// average Freq command of the last minute [ppb,10^-9]
      int32 EDStatus.F_P; // peak dF over 60sec period during the last minute
                          // [ppt,10^-12]
                          // F_P = max(abs(F_Mn-F_Av),abs(F_Mx-F_Av))
} EDStatus;
```

4.4.15.2 EDSTT

```
// External Device Status Time - Elapsed Time and Time
// xx_ED_STn: External Device Status
// 0 – Init
// 1 – Reserved
// 2 – No signal/No valid signal
// 3 – Trace
// 4 – Lock
// xx_G_STn: Global Status
// 0 – Init / No Signal
// 1 – Free run
// 2 – Holdover
// 3 – Trace
// 4 – Lock

typedef struct {
      int32 EDSTT.ET_ED_ST0;
      int32 EDSTT.ET_ED_ST1;
      int32 EDSTT.ET_ED_ST2;
      int32 EDSTT.ET_ED_ST3;
      int32 EDSTT.ET_ED_ST4;

      int32 EDSTT.T_ED_ST0;          // PTP time
      int32 EDSTT.T_ED_ST1;          // PTP time
      int32 EDSTT.T_ED_ST2;          // PTP time
      int32 EDSTT.T_ED_ST3;          // PTP time
      int32 EDSTT.T_ED_ST4;          // PTP time
```

```
      int32 EDSTT.ET_G_ST0;
      int32 EDSTT.ET_G_ST1;
      int32 EDSTT.ET_G_ST2;
      int32 EDSTT.ET_G_ST3;
      int32 EDSTT.ET_G_ST4;

      int32 EDSTT.T_G_ST0;          // PTP time
      int32 EDSTT.T_G_ST1;          // PTP time
      int32 EDSTT.T_G_ST2;          // PTP time
      int32 EDSTT.T_G_ST3;          // PTP time
      int32 EDSTT.T_G_ST4;          // PTP time
} EDSTT;
```

4.4.15.3 AdiState

```
typedef struct AdiState {
      int32 adiRefClkState;
      int32 adiDpllState;
} AdiState;
```

4.4.16 Other

4.4.16.1 clockClassPar

```
typedef struct clockClassPar {
      u_char DefCcTr;
      u_char DefCcLk;
} clockClassPar;
```

5 Application Programming Interface (API) Functions

5.1 General

Controlling, monitoring or configuring the device can be done using API functions. The API functions library is provided by IPClock and can be used by the host.

Each API function is described in the following manner:

Description	Short description of the API function **`API function prototype`**
Applicability	Applicable to either/or the IPC9xxx, IPC17xx, IPC1603
Example	Example using the API function
Parameters	Parameters passed and returned by the API function. N/A in range is indicating either no range limitation or that the function is returning a structure.

The API function return a code indicating whether the command was completed successfully or it failed. Table 3 below describes the returned **STATUS code.**

STATUS	Description
0	API Success.
-1	API Error. Error with API use or parameters.
-2	API Error. Management port is busy.
-3	API Error. No such API.
-4	API Error. Device CRC error. Device detect CRC error.
-5	API Error. Host CRC error. Host driver detect CRC error.
-6	API Error. Host timeout error. Host driver does not get API response.
-7	API return empty. No information.

Table 3: Returned STATUS Code

It is recommended to use the device's APIs while the device is ready for operation, as indicated by:

- While at one of the following states - free run (FR), holdover (HO), trace (TR), lock (LK). For detailed information about the device states, see the following chapter.
- Or after the device was reported the "…. is ready" message (e.g. [#0113] IPC9000 T-GM is ready).

5.2 Device State API Functions

5.2.1 General

5.2.1.1 Slave and BC Modes

5.2.1.1.1 Device State-Machine

Figure 2 below depicts the device state-machine when in slave and BC modes.

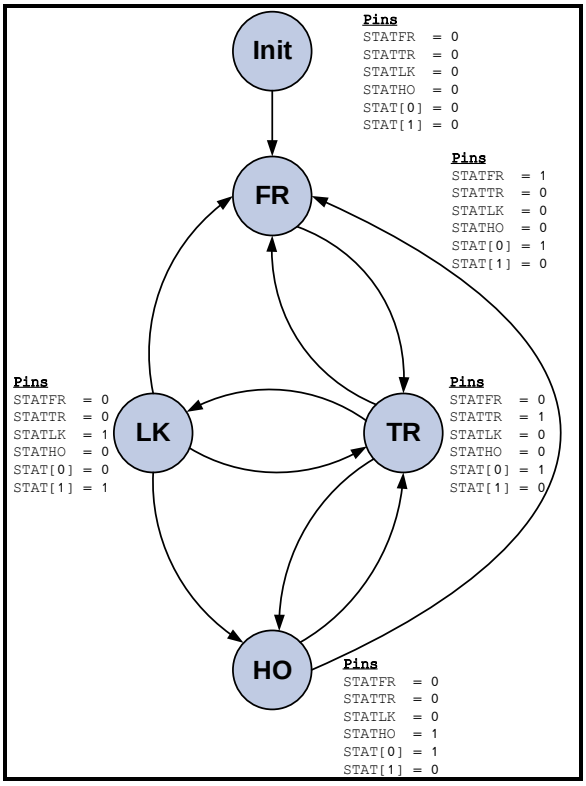

Figure 2: Device State-Machine – Slave and BC Modes

The device starts in Init state. In this state, the device is preparing for operating. Once completed, the device will transition to free run (FR) state. The device will transition to trace (TR) state when a valid IEEE 1588 packet stream is detected by the slave port.

In TR state, the device is waiting for the slave port to attempt locking to the master. If successful, the device will transition to lock (LK) state. In case the device will be unable to lock to the master within 120 minutes, it will transition to free run (FR) state.

In case the slave port lost the IEEE 1588 packet stream from its master, and the device is not in holdover ready state, the device will transition to free run (FR) either from LK state or from TR state. In case the device is in holdover ready state, the device will transition to holdover (HO) either from LK state or from TR state. The device will transition from HO to FR state in case the slave port was in HO state for a consecutive 120 minutes or in accordance with HoSepcMode. The device will transition to TR state should the packet stream resumes to the slave port.

Valid IEEE 1588 packet stream is defined using the Communication path (Commpath) term. The Commpath, is the PTP Sync/FU and DelayResp packets path between a master port and a slave port. The slave Commpath state (slaveCommpathState) indicates the current Commpath as been monitored by the slave port.

Up Commpath state (slaveCommpathState=1) indicates that the slave is receiving both Sync/FU packets and DelayResp packets, as expected. The transition to up Commpath state is pending detection of sequential series of 15 Sync/FU packets followed by one sec duration with one delay response packet or more.

Down Commpath state (slaveCommpathState=0) indicates that the slave is neither receiving Sync packets nor DelayResp Packets. The Commpath is down in case:

- No Sync packets for more than 1 sec

- No DelayResp packets for more than 30 sec.

STATFR, STATTR, STATLK, STATHO and STAT[1:0] pins are indicating the device state and status. See datasheet for more details regarding those pins.

The table below summarizes the state transitions events.

Transition	Event
➔ Init	Power up.
Init ➔ FR	Initialization completed and BIST pass.
FR ➔ TR	***slaveCommpathState=1*** as reported by ***GetIpcPtpCommpathState***.
TR ➔ LK	***slaveCommpathState=1*** as reported by ***GetIpcPtpCommpathState*** and estimated phase error is within the defined threshold. The estimated phase error presented by ***slaveStateEx*** as reported by ***GetIpcPtpState*** API. The transition is determined by ***slaveStateEx*** and by ***slaveStateLkTh***, set by ***SetIpcPtpStateSlaveLkTh*** API.
TR ➔ HO	***slaveCommpathState=0*** as reported by ***GetIpcPtpCommpathState***, while ***slaveStateHoReady*** as reported by ***GetIpcPtpStateSlaveHoReady*** is greater than 0.
LK ➔ TR	***slaveCommpathState=1*** as reported by ***GetIpcPtpCommpathState***, and estimated phase error is exceeding the defined threshold. The estimated phase error presented by ***slaveStateEx*** as reported by ***GetIpcPtpState*** API. The transition is determined by ***slaveStateEx*** and by ***slaveStateLkTh***, set by ***SetIpcPtpStateSlaveLkTh*** API.
LK ➔ HO	***slaveCommpathState=0*** as reported by ***GetIpcPtpCommpathState***, while ***slaveStateHoReady*** as reported by ***GetIpcPtpStateSlaveHoReady*** is greater than 0.
LK ➔ FR	***slaveCommpathState=0*** as reported by ***GetIpcPtpCommpathState***, while ***slaveStateHoReady*** as reported by ***GetIpcPtpStateSlaveHoReady*** is 0.
HO ➔ TR	***slaveCommpathState=1*** as reported by ***GetIpcPtpCommpathState***.
HO ➔ FR	Consecutive 120min in HO state or in accordance with ***HoSepcMode*** as set by ***SetIpcPtpAdminHoSpecMode.***
TR ➔ FR	***slaveCommpathState=1*** as reported by ***GetIpcPtpCommpathState***, while ***slaveStateHoReady*** as reported by ***GetIpcPtpStateSlaveHoReady*** is 0, or in TR state for more than consecutive 120min.

Table 4: Device State Transition Events Table – Slave and BC Modes

5.2.1.1.2 Device State

The device state is reported by API GetIpcPtpState (return gState.state). The state can be Init, Free run, Holdover, Trace and Lock (0 to 4 respectively). The STATFR, STATTR, STATLK and STATHO pins present the device state as well. For additional information about those pins, refer to the data sheet.

In Slave and BC modes the gState.state resolution is been performed using gState.slaveStateEx, reported by API GetIpcPtpState, and by slaveStateLkTh threshold sets by API SetIpcPtpStateSlaveLkTh. The resolution logic for getting gState.state as a function of gState.slaveStateEx and slaveStateLkTh is shown in Table 5.

gState.slaveStateEx		**gState.state**	
Value	**Description**	**Value**	**Description**
0	Init	0	Init
1	Free run	1	Free run
2	Holdover	2	Holdover
3	100µsec < |Err|	3	Trace
4	10µsec < |Err| ≤ 100µsec	3 or 4	Trace or Lock
5	5µsec < |Err| ≤ 10µsec	3 or 4	Trace or Lock
6	1µsec < |Err| ≤ 5µsec	3 or 4	Trace or Lock
7	|Err| ≤ 1µsec	3 or 4	Trace or Lock
8	|Err| ≤ 1µsec & FFOFF≤40ppb	4	Trace or Lock

Table 5: gState.state Resolution

The gState.state resolution Logic:

```
IF        (slaveStateLkTh <= gState.slaveStateEx)        THEN gState.state = Lock
ELSEIF (3 < gState.slaveStateEx < slaveStateLkTh)        THEN gState.state = Trace
ELSEIF (3 = gState.slaveStateEx)                         THEN gState.state = Trace
ELSEIF (2 = gState.slaveStateEx)                         THEN gState.state = Holdover
ELSEIF (1 = gState.slaveStateEx)                         THEN gState.state = Free run
ELSE                                                     THEN gState.state = Init  // gState.slaveStateEx=0
ENDIF
```

By default, slaveStateLkTh=6, thus the slave state is Lock upon time error smaller or equal to 5μsec. In tha case the gState.state resolution is shown in Table 6.

gState.slaveStateEx		**gState.state**	
Value	**Description**	**Value**	**Description**
0	Init	0	Init
1	Free run	1	Free run
2	Holdover	2	Holdover
3	100μsec < |Err|	3	Trace
4	10μsec < |Err| ≤ 100μsec	3	Trace or Lock
5	5μsec < |Err| ≤ 10μsec	3	Trace or Lock
6	1μsec < |Err| ≤ 5μsec	4	Trace or Lock
7	|Err| ≤ 1μsec	4	Trace or Lock
8	|Err| ≤ 1μsec & FFOFF≤40ppb	4	Trace or Lock

Table 6: gState.state Resolution for slaveStateLkTh=6

In Master mode the gState.state is defined by the master PLL. The master PLL is responsible to lock to the clock signal available in CLKIN pin. For additional information about CLKIN pin, refer to the datasheet. In Master mode the slaveStateEx returns 0.

5.2.1.2 Master Mode

5.2.1.2.1 Device State-Machine

Figure 3 below depicts the device synchronization state-machine when in master mode.

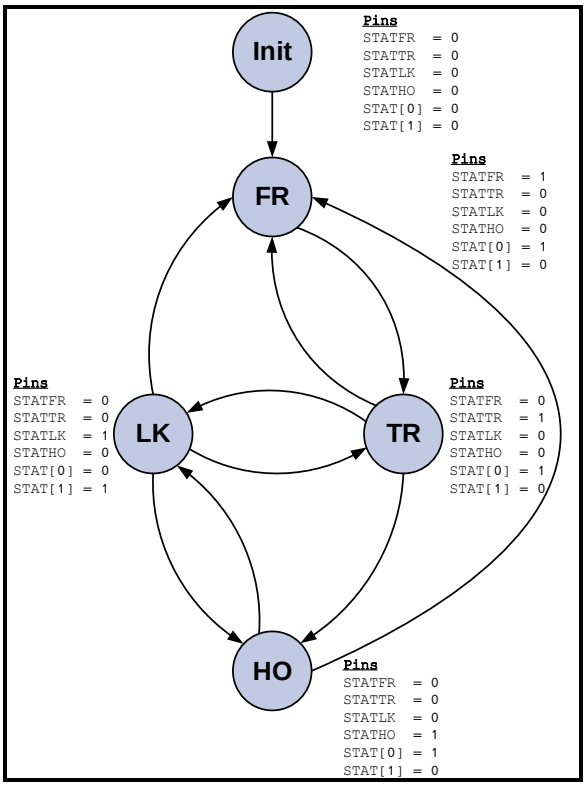

Figure 3: Device State-Machine – Master Mode

The device starts in Init state. In this state, the device is preparing for operating. Once completed, the device will transition to free run (FR) state. The device will transition to trace (TR) state when a 1PPS signal or reference clock is detected (according to configuration).

In TR state, the device is waiting for the master port to attempt locking to the 1PPS signal or reference clock. If successful, the device will transition to lock (LK) state. In case the device will be unable to lock within 120 minutes, it will transition to free run (FR) state.

In case the master port lost its 1PPS signal or reference clock, and the device was never been in LK state, the device will transition to free run (FR) either from LK state or from TR state. In case the device was in LK state, the device will transition to holdover (HO) either from LK state or from TR state. The device will transition from HO to FR state in case the master port was in HO state for a consecutive 120 minutes or in accordance with HoSpecMode. In case device in HO, and 1PPS signal or reference clock resumed and it is aligned with the device internal clock, the device will transition from HO to LK. In case of 1PPS signal or reference clock resumed and is not aligned with the device internal clock, the device will transition from HO to FR then TR, and in case lock conditions met, will transition to LK.

STATFR, STATTR, STATLK, STATHO and STAT[1:0] pins are indicating the device state and status. See datasheet for more details regarding those pins.

The table below summarizes the state transitions events.

Transition	Event
➔ Init	Power up
Init ➔ FR	Initialization completed and BIST pass.
FR ➔ TR	1PPS signal or reference detected
TR ➔ LK	Estimated phase synchronization error estimated ≤100ns
TR ➔ HO	1PPS signal or reference clock is lost
LK ➔ TR	Estimated phase synchronization error estimated ≤100ns
LK ➔ HO	1PPS signal or reference clock is lost
HO ➔ LK	1PPS signal or reference clock resumed and estimated phase synchronization error estimated ≤100ns
HO ➔ FR	Consecutive 120min in HO state or in accordance with ***HoSpecMode*** as set by ***SetIpcPtpAdminHoSpecMode.*** Or in case 1PPS signal or reference clock resumed and is not aligned with the device internal clock
TR ➔ FR	Consecutive 120min in TR state

Table 7: Device State Transition Events Table – Master Mode

5.2.1.3 Device Status

The device status is reported by API GetIpcPtpState (return gState.status). The status can be PASSED, FAILED and ALARMED (1 to 3 respectively). The STAT[1:0] pins present the device status as well. For additional information about STAT[1:0] pins, refer to the data sheet.
The gState.status is function of two items: device state and the last BIST.

The last known BIST, performed automatically by the device or by the user application, and the device state, are been used for the gState.status resolution. The logic used for gState.status resolution describes in Table 8.

Last BIST Result	gState.state	gState.status
FAILED	Init	FAILED
FAILED	Free run	FAILED
FAILED	Holdover	FAILED
FAILED	Trace	FAILED
FAILED	Lock	FAILED
ALARMED	Init	ALARMED
ALARMED	Free run	ALARMED
ALARMED	Holdover	ALARMED
ALARMED	Trace	ALARMED
ALARMED	Lock	ALARMED
PASSED	Init	ALARMED
PASSED	Free run	ALARMED
PASSED	Holdover	ALARMED

PASSED	Trace	ALARMED
PASSED	Lock	PASSED

Table 8: gState.status Resolution

Typically right after power up, gState.status is 9 indicates the device was reset (STAT[1:0]=[0,0]=FAIL), then gState.status is ALARMED (STAT[1:0]=[0,1]=ALARMED). Following gState.state in Lock, gState.status will become PASSED (STAT[1:0]=[1,0]=PASSED).

5.2.1.4 BIST - Built in Self Test

The device BIST monitor the device operation starting during Initialization phase and continue during Showtime phase.

As part of power-up (or SetIpcPtpAdminStop/SetIpcPtpAdminStart/SetIpcPtpAdminReset) Initialization BIST is been performed. The BIST is performed according to the operation mode (Master/Slave/BC). The BIST results are avalible as follow:

- In Master mode GetIpcAdminBistRepMasterInit
- In Slave mode GetIpcAdminBistRepSlaveInit
- In BC mode GetIpcAdminBistRepSlaveInit and GetIpcAdminBistRepSlaveInit

Later on, during device operation the Showtime BIST is been performed. The BIST is performed according to the operation mode (Master/Slave/BC). The BIST results are avalible as follow:

- In Master mode GetIpcAdminBistRepMasterShowtime
- In Slave mode GetIpcAdminBistRepSlaveShowtime
- In BC mode GetIpcAdminBistRepMasterShowtime and GetIpcAdminBistRepSlaveShowtime

The BIST elements are defined as follow:
typedef u_char TBistTestResult_t;

1 – PASSED
2 – FAILED
3 – ALARMED

The BIST is been performed automatically by the device.

For more information about BIST, refer to the following chapters:
-- Chapter “BIST – Built-in Self Test” - APIs
-- Chapter “Power Up & Start Operation Reports” – power up log reports
-- Chapter “Built In Self Test (BIST)” – BIST description

5.2.2 GetIpcPtpState

Description	Get device state.
	`STATUS GetIpcPtpState(`
	`  OUT GState* gState);`			

Applicability	IPC9xxx, IPC17xx, IPC1603			
Example	rv=GetIpcPtpState(&gState)			
Parameters	Parameter Name		Range	Default
	status	0	FAIL – Device is not operating as required	0
		1	ALARM – Device may not operating as required or it is not in lock state	
		2	PASS – Device is operating as required and it is in lock state	
		9	Device was Reset (Auto clear)	
	state	0	Init	0
		1	Free run	
		2	Holdover	
		3	Trace	
		4	Lock	
	slaveStateEx	0	Init	0
		1	Free run	
		2	Holdover	

	3	100µsec < |Err|
	4	10µsec < |Err| ≤ 100µsec
	5	5µsec < |Err| ≤ 10µsec
	6	1µsec < |Err| ≤ 5µsec
	7	|Err| ≤ 1µsec
	8	|Err| ≤ 1µsec and FFOFF≤40ppb

Note ***status*** *indication: identical to STAT[1:0] pins. For additional information on STAT[1:0], refer to the device data sheet.*

Note ***slaveStateEx*** *applicable in BC and slave modes. In Master mode* ***slaveStateEx*** *returns 0.*

5.2.3 GetIpcPtpStateMasterAll

Description: Get the current state. Include the elapsed time and the PTP time of the current state.

```
STATUS GetIpcPtpStateMasterAll(
        IN int32 fnId,
        OUT ManagerState* masterStateAll);
```

Applicability: IPC9xxx – Master mode, IPC17xx – Master mode

Example: rv=GetIpcPtpStateMasterAll(0,&masterStateAll)

Parameters:

Parameter Name	Range	Default
fnId	0	0
masterStateAll	0	N/A

5.2.4 SetIpcPtpStateSlaveLkTh

Description: Set the lock indication threshold while in Slave mode. The Slave shall transit from Trace state to Lock state upon estimating the time error to be within slaveStateLkTh threshold set. By default, the slave state is Lock upon time error smaller or equal to 5µsec. slaveStateEx

```
STATUS SetIpcPtpStateSlaveLkTh(
    IN int32 fnId,
    IN int32 slaveStateLkTh);
```

Applicability: IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603

Example: rv=SetIpcPtpStateSlaveLkTh(0,5).
The Slave shall transit from Trace state to Lock state if the time error is less or equal to 10µsec.

Parameters:

Parameter Name		Range	Default
fnId	0		0
slaveStateLkTh	4	|Err| ≤ 100µsec	6
	5	|Err| ≤ 10µsec	
	6	|Err| ≤ 5µsec	
	7	|Err| ≤ 1µsec	
	8	|Err| ≤ 1µsec and FFOFF≤40ppb	

5.2.5 GetIpcPtpStateSlaveLkTh

Description: Get the lock indication threshold while in Slave mode.

```
STATUS GetIpcPtpStateSlaveLkTh(
    IN int32 fnId,
    OUT int32* slaveStateLkTh);
```

Applicability: IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603

Example: rv=GetIpcPtpStateSlaveLkTh(0,&slaveStateLkTh)

Parameters:

Parameter Name	Range	Default
fnId	0	0
slaveStateLkTh	See SetIpcPtpStateSlaveLkTh	6

5.2.6 GetIpcPtpStateSlaveAll

Description: Get the current state. Include the elapsed time and the PTP time of the current

	state.
	`STATUS GetIpcPtpStateSlaveAll(`
	`IN int32 fnId,`
	`OUT SlaveManagerState* slaveStateAll);`

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=GetIpcPtpStateSlaveAll(0,&slaveStateAll)

Parameters

Parameter Name	Range	Default
fnId	0	0
slaveStateAll	N/A	N/A

5.2.7 GetIpcPtpStateSlaveHoReady

Description	Get the slaveStateHoReady status (Holdover ready). The slaveStateHoReady turn to 1 when the device's servo reach the state which enables to move into holdover state (HO) in case connection to master port lost. The slaveStateHoReady turn to 2 when the device's servo reach to the HO deep ready state which enables to move into HO state with enhanced stability. The slaveStateHoReady turn to 0 after being 1 or 2 in case device in HO state for more than 2 hours or in TR (Trace) for more than 1 hour.
	`STATUS GetIpcPtpStateSlaveHoReady(`
	`IN int32 fnId,`
	`OUT int32* slaveStateHoReady);`

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=GetIpcPtpStateSlaveHoReady(0,&slaveStateHoReady)

Parameters

Parameter Name		Range	Default
fnId	0		0
slaveStateHoReady	0	HO not ready state	0
	1	HO ready state	
	2	HO deep ready state	

Note *The preconditions for the device to get into slaveStateHoReady=1 are the logical OR of the following:*
(1) Was or is at slaveStateEx=8
(2) Device in LK for more than 30 minutes

Note *The preconditions for the device to get into slaveStateHoReady=2 are logical AND of the following:*
(1) slaveStateHoReady=1
(2) Extremely good network conditions (e.g. PDV<1us)

5.3 PTP API Functions

5.3.1 Admin

5.3.1.1 GetIpcAdminVer

Description	Get product version.
	`STATUS GetIpcAdminVer(`
	`IN int32 fnId,`
	`OUT char** Ver);`

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=GetIpcAdminVer(0,&ver)

Parameters

Parameter Name	Range	Default
fnId	0	0
ver	N/A	N/A

5.3.1.2 SetIpcAdminStore

Description	Store the configuration database (configDB) on the local FLASH memory.
	`STATUS SetIpcAdminStore(`
	`IN int32 fnId,`
	`IN int32 configNo);`

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=SetIpcAdminStore(0,0)

Parameters

Parameter Name	Range	Default

fnId	0	0
configNo	0 99 – reset/erase configDB	0

The configuration database (configDB) includes the parameters that are required for either boundary clock, master or slave clocks to return to the same settings. For example, network interface parameters like IP address and MAC address, mode of operation and packet rate are all stored in configDB. Those parameters are been used when device starts to work due to power up the device, or after device reset using SetIpcPtpAdminReset, or after using SetIpcPtpAdminStop / SetIpcPtpAdminStart APIs. The configDB includes the parameters relevant to the mode of operation at the time the SetIpcAdminStore was invoked. As examples, if SetIpcAdminStore was invoked when the IPC9xxx is in BC mode then the IPC9xxx shall operate as BC with relevant parameters to BC operation, if SetIpcAdminStore was invoked when the IPC9xxx is in master mode, then the IPC9xxx shall operate as master with relevant parameters to master operation.

Following by using SetIpcAdminStore(0,0):
-- Performing device's power up or device's reset - leads the device to use the entire stored configuration (e.g. startMode, ifcNetIpAddr, ifcNetMacAddr, ptpClockClass, ucastMasterTable).
-- Performing device SetIpcPtpAdminStop and SetIpcPtpAdminStart using the same startMode - leads the device to use the entire stored configuration (e.g. startMode, ifcNetIpAddr, ifcNetMacAddr, ptpClockClass, ucastMasterTable).
-- Performing device SetIpcPtpAdminStop and SetIpcPtpAdminStart using different startMode - leads the device to use ONLY the network stored configuration (ifcNetIpAddr, ifcNetMacAddr, ifcNetIpSubnetMask, ifcNetIpDefaultGateway). Other parameters will be set to their default values (e.g. ptpClockClass, ucastMasterTable).

Following by using SetIpcAdminStore(0,99), the default configuration is been used ONLY after device's power up or device's reset or performing SetIpcPtpAdminStop and SetIpcPtpAdminStart:
-- Performing device's power up or device's reset - leads the device to use default configuration (e.g. startMode, ifcNetIpAddr, ifcNetMacAddr, ptpClockClass, ucastMasterTable).
-- Performing SetIpcPtpAdminStop and SetIpcPtpAdminStart - leads the device to use the default configuration and SetIpcPtpAdminStart parameters (startMode, pktRate, extDevMode, portMapMode).
In order to fully clear the confDB the following sequence shall be done:
-- SetIpcAdminStore(0,99)
-- SetIpcPtpAdminReset / SetIpcPtpAdminStop-SetIpcPtpAdminStart / hardware reset

The SetIpcPtpAdminStartPar requires SetIpcAdminStore. The SetIpcPtpAdminStartPar sets the mode of operation parameters (e.g. startMode) that will be used after device power cycle or after using SetIpcPtpAdminReset. This mode of operation maybe different than the device current mode of operation. As an example, startMode=0 may be the current mode, while the user would like that startMode=4 will be used after the next power cycle or after SetIpcPtpAdminReset. In order to enforce parameters changings, SetIpcAdminStore shall be used after SetIpcPtpAdminStartPar.

SetIpcAdminStore(0,0) can be used only while device is ready for operation. This is done in order to verify the device has a valid configuration before storing it. Valid configuration includes startMode as set by SetIpcPtpAdminStart. Thus SetIpcAdminStore(0,0) can be done only after one of the "IPCxxxx … is ready" messages was displayed (e.g. "IPCxxxx Unicast Slave is ready").

In case device was stopped by SetIpcPtpAdminStop. SetIpcAdminStore(0,0) cannot be used.

SetIpcAdminStore(0,99) can be used at any stage.

5.3.1.3 SetIpcPtpAdminStart

Description	The SetIpcPtpAdminStart sets the mode of operation. **STATUS SetIpcPtpAdminStart(** **IN int32 fnId,** **IN int32 startMode,** **IN int32 pktRate,** **IN int32 extDevMode,** **IN int32 portMapMode); [P64]**
Applicability	IPC9xxx, IPC17xx, IPC1603
Example	IPC17xx/IPC9xxx - unicast Master mode: rv=SetIpcPtpAdminStart(0,0,128,0,0) IPC17xx/IPC9xxx - unicast Slave mode requesting Sync packet rate 32pps: rv=SetIpcPtpAdminStart(0,1,32,0,0) IPC17xx/IPC9xxx - unicast Slave mode requesting Sync packet rate 128pps:

rv=SetIpcPtpAdminStart(0,1,128,0,0)
IPC9xxx in BC mode requesting Sync packet rate of 32pps:
rv=SetIpcPtpAdminStart(0,4,32,0,0)
IPC9600 - unicast Master mode, No external device:
rv=SetIpcPtpAdminStart(0,0,128,0,0)
IPC9xxx in BC mode requesting Sync packet rate of 32pps, distributed: [P64]
rv=SetIpcPtpAdminStart(0,4,32,0,1)
IPC9600 - unicast Master mode, External device 4, mode B:
rv=SetIpcPtpAdminStart(0,0,128,44,0)
IPC9600 - unicast Slave mode, Sync packet rate 32pps, External device 4, mode B:
rv=SetIpcPtpAdminStart(0,1,32,42,0)
IPC9600 - unicast Slave mode, Sync packet rate 128pps, External device 4, mode B:
rv=SetIpcPtpAdminStart(0,1,128,42,0)
IPC9600 in BC mode, Sync packet rate of 32pps, External device 4, mode A:
rv=SetIpcPtpAdminStart(0,4,32,41,0)

Parameters

Parameter Name	Range		Default	
fnId	0		0	
startMode	0	Unicast[1][4] OC-Master	IPC17xx/1603	1
	1	Unicast[1] OC-Slave	IPC9xxx	4
	2	Multicast/Unicast[2][4][14] OC-Master		
	3	Multicast/Unicast[2][14] OC-Slave		
	4	Unicast BC[1][3][12]		
	7	Multicast[4][5][14] OC-Master		
	8	Multicast[5][14] OC-Slave		
	10	Multicast/Unicast-B[2][6][14] OC-Slave		
	22	T-GM: Telecom Grandmaster (ITU-T G.8275.1).[8][9][13]		
	23	T-TSC: Telecom Time Slave Clock (ITU-T G.8275.1).[8][9][13]		
	24	T-BC: Telecom BC (ITU-T G.8275.1).[8][9][12][13]		
pktRate[4]	startMode=0-10			
	2^N (N=1…5)	BC	BC	32
	2^N (N=1…7)	OC-Slave	OC-Slave	32
	128	OC-Master	OC-Master	128
	startMode=22-24			
	2^4 (N=4)	T-BC	T-BC	16
	2^4 (N=4)	T-TSC	T-TSC	16
	128	T-GM	T-GM	128
extDevMode[7]	0	No external device	0	
	1	Reserved		
	2	Reserved		
	3	Reserved		
	4x	External device 4 (ED4)		
	5	Reserved		
portMapMode[10][11]	0	Centralized – No mapping	0 [P64]	
	1	Distributed – Mapping physical ports		
	2	Distributed A – Default A of acceptableTable		
	3	Distributed – Empty acceptableTable		
	4	Distributed – Mapping using acceptableTable & SrcMac insertion (startMode=22-24)		

[1] All 1588 v2 packets are sent as Unicast. PTP default profile (IEEE 1588-2008, Annex J - chapter: J.3 Delay Request-Response Default PTP profile).
[2] Sync and Announce packets are sent as Multicast, Delay_Req and Delay_Resp as Unicast.
[3] Applicable only for IPC9xxx.
[4] In Master, (Unicast and Multicast modes) the pktRate is the maximal Sync packet rate transmitted by the Master. When invoking Master, the pktRate MUST be set to 128. In startMode=2,7 the default Sync packet rate is 32 packets per sec. In startMode=22,24 the

default Sync packet rate is 16 packets per sec. In order to change the Sync packet rate, the SetIpcPtpMcastSyncRate API shall be used.
(5) Announce, Sync, Delay_Req and Delay_Resp packets are sent as Multicast.
(6) Slave does not send RUTS (REQUEST UNICAST TRANSMISSION TLV) packets for Unicast Delay_Resp.
(7) External Device (PLL) Mode. Applicable only for IPC9600.
(8) PTP telecom profile for time/phase synchronization (ITU-T G.8275.1) – named as profile-2 in the user guide. ITU-T G.8275.1 is applicable for IPC9xxx and IPC17xx only.
(9) Sync packet rate is 16 packets per sec.
(10) portMapMode=1,2,3 applicable only for IPC9xxx & IPC17xx, startMode=0,1,4,22,23,24. portMapMode=4 applicable only for IPC9xxx & IPC17xx, startMode=22,23,24
(11) Distributed: Based on (1) PortNums & AcceptableTable (2) PortNums. [P64].
(12) startMode = 4,24 do not applicable for IPC17xx.
(13) The following MAC address are been supported:
01-80-C2-00-00-0E - as per G.8275.1. startMode = 22,23,24.
01-1B-19-00-00-00 - as per G.8275.1. startMode = 22,23,24.
(14) The following MAC address is been supported:
01-00-5E-00-01-81 - as per IEEE 1588. startMode = 2,3,7,8,10. MAC address of 224.0.1.129.

The table below describes the extDevMode configuration and the supported combinations of startMode and extDevMode.

extDevMode	*Operation Mode*	*AD9548 Configuration*	*startMode*		
			Master	*Slave*	*BC*
			0,2,7	*1,3,8,10*	*4*
40	*ED4 init*	*SetIpcAdiConf0*	+	+	+
41	*ED4 mode A*	*SetIpcAdiConfA*		+	+
42	*ED4 mode C*	*SetIpcAdiConfC*		+	+
43	*ED4 mode D*	*SetIpcAdiConfD*		+	+
44	*ED4 mode B*	*SetIpcAdiConfB*	+		
45	*ED4 mode E*	*SetIpcAdiConfE*		+	+
46	*ED4 init*	*SetIpcAdiConf0 & DDS control (e.g. extDevMode = 41)*		+	+
47	*ED4 mode F*	*SetIpcAdiConfF*		+	+

portMapMode=0: Centralized / No mapping - 2 logical ports, unlimited number of physical ports.
portMapMode=1: Distributed / Mapping - Multiple logical ports, multiple physical ports, one-to-one mapping. See GetIpcPtpAcceptableTable for default acceptableTable.
portMapMode=2: Distributed / Mapping A - Multiple logical ports, multiple physical ports, one-to-one mapping. See GetIpcPtpAcceptableTable for default A acceptableTable.
portMapMode=3: Distributed / Mapping A - Multiple logical ports, multiple physical ports, one-to-one mapping. Empty acceptableTable.
portMapMode=4: Distributed / Mapping & Insert using acceptableTable & SrcMac - Multiple logical ports, multiple physical ports, one-to-one mapping using acceptableTable & insertion SrcMac to all ingress packets. clockIdentity shall be in accordance with IEEE EUI-48 as per IEEE 1588 standard. Empty acceptableTable.

5.3.1.4 SetIpcPtpAdminStop

Description	The SetIpcPtpAdminStop stops the PTP operation.
	`STATUS SetIpcPtpAdminStop(`
	`  IN int32 fnId);`		

Applicability	IPC9xxx, IPC17xx, IPC1603		
Example	rv=SetIpcPtpAdminStop(0)		
Parameters	Parameter Name	Range	Default
	fnId	0	0

5.3.1.5 GetIpcPtpAdminRole

Description	The GetIpcPtpAdminRole shows the mode selected by SetIpcPtpAdminStart.
	`STATUS GetIpcPtpAdminRole(`
	`  IN int32 fnId,`
	`  OUT int32* role);`		

Applicability	IPC9xxx, IPC17xx, IPC1603		
Example	rv=GetIpcPtpAdminRole(0,&ptpRole)		
Parameters	Parameter Name	Range	Default

fnId	0		0
role	See	SetIpcPtpAdminStart	IPC17xx/1603 1 IPC9xxx 4

Note *For additional information, refer to SetIpcPtpAdminStart.*

5.3.1.6 GetIpcAdminExtDeviceMode

Description: The GetIpcAdminExtDeviceMode shows the extDevMode selected by SetIpcPtpAdminStart.

```
STATUS GetIpcAdminExtDeviceMode(
    IN int32 fnId,
    OUT int32* extDevMode);
```

Applicability: IPC9600

Example: rv=GetIpcAdminExtDeviceMode(0,&extDevMode)

Parameters:

Parameter Name		Range	Default
fnId	0		0
extDevMode	0	No external device	0
	1	Reserved	
	2	Reserved	
	3	Reserved	
	4x	External device 4	
	5	Reserved	

5.3.1.7 SetIpcPtpAdminReset

Description: Resets the device.

```
STATUS SetIpcPtpAdminReset(
    IN int32 fnId);
```

Applicability: IPC9xxx, IPC17xx

Example: rv=SetIpcPtpAdminReset(0)

Parameters:

Parameter Name		Range	Default
fnId	0		0

5.3.1.8 SetIpcPtpAdminStartPar

Description: The SetIpcPtpAdminStartPar sets the mode of operation parameters (startMode, pktRate, extDevMode, portMapMode) that will be used after device power cycle or after using SetIpcPtpAdminReset. In order to enforce parameters changings, SetIpcAdminStore shall be used after SetIpcPtpAdminStartPar.

```
STATUS SetIpcPtpAdminStartPar(
    IN int32 fnId,
    IN int32 startMode,
    IN int32 pktRate,
    IN int32 extDevMode,
    IN int32 portMapMode);
```

Applicability: IPC9xxx, IPC17xx, IPC1603

Example: See SetIpcPtpAdminStart
Ignore: rv=SetIpcPtpAdminStartPar(0,-1,-1,-1,-1)

Parameters:

Parameter Name		Range	Default
fnId	0		0
startMode	-1	Ignore	-1
	See	SetIpcPtpAdminStart	
pktRate	-1	Ignore	-1
	See	SetIpcPtpAdminStart	
extDevMode	-1	Ignore	-1
	See	SetIpcPtpAdminStart	
portMapMode	-1	Ignore	-1
	See	SetIpcPtpAdminStart	

For additional information, refer to SetIpcPtpAdminStart.

(1) This API is applicable only if FLASH memory is used.
(2) The new parameters settings will take effect only after storing configDB in FLASH memory and device reset.

The SetIpcPtpAdminStartPar requires SetIpcAdminStore. The SetIpcPtpAdminStartPar sets the mode of operation parameters (e.g. startMode) that will be used after device power cycle or after using SetIpcPtpAdminReset. This mode of operation maybe different than the device current mode of operation (e.g startMode=0 is the current, while startMode=4 shall be used after power cycle or SetIpcPtpAdminReset. In order to enforce parameters changings, SetIpcAdminStore shall be used after SetIpcPtpAdminStartPar.

5.3.1.9 GetIpcPtpAdminStartPar

Description	Get the startPar. In case SetIpcPtpAdminStartPar was not used, the API will return parameters with value of -1.
	`STATUS GetIpcPtpAdminStartPar(`
	`  IN int32 fnId,`
	`  OUT StartPar* startPar);`		

Applicability	IPC9xxx, IPC17xx, IPC1603		
Example	rv=GetIpcPtpAdminStartPar(0,&startPar)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	startPar	See SetIpcPtpAdminStartPar	N/A

5.3.1.10 SetIpcPtpAdminManId

Description	Set manufacturer identity.
	`STATUS SetIpcPtpAdminManId(`
	`  IN int32 fnId,`
	`  IN const Octet* manId);`		

Applicability	IPC9xxx, IPC17xx, IPC1603		
Example	rv=SetIpcPtpAdminManId(0,”9ABCDE”)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	manId	N/A	FFFFFF

5.3.1.11 GetIpcPtpAdminManId

Description	Get manufacturer identity.
	`STATUS GetIpcPtpAdminManId(`
	`  IN int32 fnId,`
	`  OUT Octet manId);`		

Applicability	IPC9xxx, IPC17xx, IPC1603		
Example	rv=GetIpcPtpAdminManId(0,&manId)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	manId	N/A	FFFFFF

5.3.1.12 SetIpcPtpAdminProdDesc

Description	Set product description.
	`STATUS SetIpcPtpAdminProdDesc(`
	`  IN int32 fnId,`
	`  IN const char* prodDesc);`		

Applicability	IPC9xxx, IPC17xx, IPC1603		
Example	rv=SetIpcPtpAdminProdDesc(0,"IPCLOCK;IPC9000;1234")		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	prodDesc	N/A	IPCLOCK;IPC9000;

5.3.1.13 GetIpcPtpAdminProdDesc

Description	Get product description.
	`STATUS GetIpcPtpAdminProdDesc(`
	`  IN int32 fnId,`
	`  OUT PTPText64* prodDesc);`

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=GetIpcPtpAdminProdDesc(0,&prodDesc)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	prodDesc	N/A	IPCLOCK;IPC9000;

5.3.1.14 SetIpcPtpAdminUserDesc

Description	Set user description.
	`STATUS SetIpcPtpAdminUserDesc(`
	`  IN int32 fnId,`
	`  IN char* userDesc);`

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=SetIpcPtpAdminUserDesc(0,"Switch-1;Rack-2;Shelf-3")

Parameters	Parameter Name	Range	Default
	fnId	0	0
	userDesc	N/A	IPCTBD1;IPCTBD2;IPCTBD3

5.3.1.15 GetIpcPtpAdminUserDesc

Description	Get user description.
	`STATUS GetIpcPtpAdminUserDesc(`
	`  IN int32 fnId,`
	`  OUT PTPText128* userDesc);`

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=GetIpcPtpAdminUserDesc(0,&userDesc)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	userDesc	N/A	IPCTBD1;IPCTBD2;IPCTBD3

5.3.1.16 SetIpcPtpAdminRevData

Description	Set revision data.
	`STATUS SetIpcPtpAdminRevDate(`
	`  IN int32 fnId,`
	`  IN const char* revData);`

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=SetIpcPtpAdminRevData(0,"A52;B35;CCD")

Parameters	Parameter Name	Range	Default
	fnId	0	0
	revData	N/A	IPC9000-20;;

5.3.1.17 GetIpcPtpAdminRevData

Description	Get revision data.
	`STATUS GetIpcPtpAdminRevDate(`
	`  IN int32 fnId,`
	`  OUT PTPText32* revData);`

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=GetIpcPtpAdminRevData(0,&revData)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	revData	N/A	IPC9000-20;;

5.3.1.18 SetIpcPtpAdminProfile

Description	Set the PTP profile to PTP default profile (IEEE 1588-2008, Annex J - chapter: J.3

	Delay Request-Response Default PTP profile) or to PTP telecom profile for frequency synchronization (ITU-T 8265.1). **STATUS SetIpcPtpAdminProfile(** **IN int32 fnId,** **IN int32 profile);**
Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=SetIpcPtpAdminProfile(0,1)

Parameters	Parameter Name		Range	Default
	fnId	0		0
	profile	0	PTP default profile (IEEE 1588-2008, Annex J - chapter: J.3 Delay Request-Response Default PTP profile)	0
		1	PTP telecom profile for frequency synchronization (ITU-T 8265.1)	

Note *The device provides Time synchronization in all profiles.*

Note *In order to change the device settings, the following sequence shall be performed after invoking the API:*
SetIpcAdminStore
SetIpcPtpAdminStop
SetIpcPtpAdminStart
Or
SetIpcAdminStore
SetIpcPtpAdminReset

Note *In profile 1 the master port shall accept Request Unicast Transmission Signaling (RUTS) packet for announce from any clock domain.*

Note *Master, Slave and BC shall be configured to the same Profile.*

Note *profile-2 is been set automaticly by the startMode=22 - 24.*

Note *SetIpcPtpAdminProfile set the ptpDomainNumber. In profile 0 the ptpDomainNumber set to 0, in profile 1 the ptpDomainNumber set to 4, and in profile 2 the ptpDomainNumber set to 24. In case user set the ptpDomainNumber, it should be done after using the SetIpcPtpAdminProfile.*

5.3.1.19 GetIpcPtpAdminProfile

Description	Get the PTP profile. **STATUS GetIpcPtpAdminProfile(** **IN int32 fnId,** **OUT int32* profile);**
Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=GetIpcPtpAdminProfile(0,&profile)

Parameters	Parameter Name		Range	Default
	fnId	0		0
	profile	0	PTP default profile	0
		1	PTP telecom profile for frequency synchronization (ITU-T 8265.1)	
		2	PTP telecom profile for time/phase synchronization (ITU-T G.8275.1)	

5.3.1.20 SetIpcPtpAdminProfileMode

Description	Set the PTP profile-2 mode. (ITU-T G.8275.1). **STATUS SetIpcPtpAdminProfileMode(** **IN int32 fnId,** **IN int32 profileMode);**
Applicability	IPC9xxx BC/Slave modes, IPC17xx Slave mode
Example	rv=SetIpcPtpAdminProfileMode(0,1)

Parameters	Parameter Name		Range	Default
	fnId	0		0
	profileMode	0	Force Sync packet rate as per profile2 (16pps)	0
		1	Enable other Sync packet rates	

In order to change the device settings, the following sequence shall be performed after invoking the API:
SetIpcAdminStore
SetIpcPtpAdminStop
SetIpcPtpAdminStart
Or
SetIpcAdminStore
SetIpcPtpAdminReset

5.3.1.21 GetIpcPtpAdminProfileMode

Description	Get the PTP profile-2 mode.

	`STATUS GetIpcPtpAdminProfileMode(`
	`  IN int32 fnId,`
	`  OUT int32* profileMode);`

Applicability	IPC9xxx BC/Slave modes, IPC17xx Slave mode
Example	rv=GetIpcPtpAdminProfileMode(0,&profileMode)

Parameters	Parameter Name		Range	Default
	fnId	0		0
	profileMode	See	SetIpcPtpAdminProfileMode	0

5.3.1.22 GetIpcPtpAdminPortMapMode

Description	Get the portMapMode.

	`STATUS GetIpcPtpAdminPortMapMode(`
	`  IN int32 fnId,`
	`  OUT int32* portMapMode);`

Applicability	IPC9xxx BC/Master modes
Example	rv=GetIpcPtpAdminPortMapMode(0,&portMapMode)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	portMapMode	See SetIpcPtpAdminStart	0

5.3.1.23 SetIpcPtpAdminHoSpecMode

Description	Set the HoSpecMode. The HoSpecMode enable to control ptpClockClass and ptpFrequencyTraceableFlag as per G.8275.1 Table 2.

	`STATUS SetIpcPtpAdminHoSpecMode(`
	`  IN int32 fnId,`
	`  IN int32 hoSpecMode);`

Applicability	IPC9xxx, IPC17xx
Example	rv=SetIpcPtpAdminHoSpecMode(0,3)

Parameters	Parameter Name		Range	Default
	fnId	0		0
	hoSpecMode	0	Disable	0 – statMode
		1	Manual – In spec (Ignore hoSpecTh)	1,3,4,8
		2	Manual – Out of spec (Ignore hoSpecTh)	3 – startMode
		3	Auto (Use hoSpecTh)	23,24

5.3.1.24 GetIpcPtpAdminHoSpecMode

Description	Get the HoSpecMode.

	`STATUS GetIpcPtpAdminHoSpecMode(`
	`  IN int32 fnId,`
	`  OUT int32* hoSpecMode);`

Applicability	IPC9xxx, IPC17xx

Example	rv=GetIpcPtpAdminHoSpecMode(0,&hoSpecMode)			
Parameters	Parameter Name		Range	Default
	fnId	0		0
	hoSpecMode	See	SetIpcPtpAdminHoSpecMode	

5.3.1.25 SetIpcPtpAdminHoSpecTh

Description	Set the HoSpecTh. The HoSpecTh defines the time in sec within holdover specification as required for hoSpecMode=3.			
	STATUS SetIpcPtpAdminHoSpecTh(**IN int32 fnId,** **IN int32 hoSpecTh);**			
Applicability	IPC9xxx, IPC17xx			
Example	rv=SetIpcPtpAdminHoSpecTh(0,900)			
Parameters	Parameter Name		Range	Default
	fnId		0	0
	hoSpecTh		0 - 86400	900

5.3.1.26 GetIpcPtpAdminHoSpecTh

Description	Get the HoSpecTh.			
	STATUS GetIpcPtpAdminHoSpecTh(**IN int32 fnId,** **OUT int32* hoSpecTh);**			
Applicability	IPC9xxx, IPC17xx			
Example	rv=GetIpcPtpAdminHoSpecTh(0,&hoSpecTh)			
Parameters	Parameter Name		Range	Default
	fnId		0	0
	hoSpecth		See SetIpcPtpAdminHoSpecTh	900

5.3.1.27 SetIpcPtpAdminFreqSrcCategory

Description	Set the FreqSrcCat.			
	STATUS SetIpcPtpAdminFreqSrcCategory(**IN int32 fnId,** **IN int32 freqSrcCat);**			
Applicability	IPC9xxx, IPC17xx			
Example	rv=SetIpcPtpAdminFreqSrcCategory(0,0)			
Parameters	Parameter Name		Range	Default
	fnId	0		0
	freqSrcCat	0	Unkown Category	0
		1	Category 1	
		2	Category 2	
		3	Category 3	

5.3.1.28 GetIpcPtpAdminFreqSrcCategory

Description	Get the FreqSrcCat.			
	STATUS GetIpcPtpAdminFreqSrcCategory(**IN int32 fnId,** **OUT int32* freqSrcCat);**			
Applicability	IPC9xxx, IPC17xx			
Example	rv=GetIpcPtpAdminFreqSrcCategory(0,&freqSrcCat)			
Parameters	Parameter Name		Range	Default
	fnId	0		0
	freqSrcCat	See	SetIpcPtpAdminFreqSrcCategory	0

5.3.1.29 GetIpcPtpAdminProfileIdentity

Description	Get the PTP profile.

```
STATUS GetIpcPtpAdminProfileIdentity(
    IN int32 fnId,
    OUT Octet* profileId);
```

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=GetIpcPtpAdminProfileIdentity(0,&profileId)

Parameters

Parameter Name	Range		Default
fnId	0		0
profile	001B19000100	PTP default profile (profile-0)	Profile-0: 001B19000100
	0019A7000100	PTP telecom profile for frequency synchronization (ITU-T G.8265.1) (profile-1)	Profile-1: 0019A7000100
	0019A7010100	PTP telecom profile for time/phase synchronization (ITU-T G.8275.1) (profile-2)	Profile-2: 0019A7010100

5.3.1.30 SetIpcPtpAdminPortEn

Description: Set the portEn parameter. The portEn enable or disable the IEEE 1588 port. This API is applicable only for SetIpcPtpSdaMode(0,1).

```
STATUS SetIpcPtpAdminPortEn(
    IN int32 fnId,
    IN Uint16 portIndex,
    IN int32 portEn);
```

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=SetIpcPtpAdminPortEn(0,1,0)

Parameters

Parameter Name	Range		Default
fnId	0		0
portIndex	portMapMode=0		N/A
	1	BC Slave port	
	2	BC-Master port	
	1	Slave mode	
	2	Master mode	
	portMapMode=>1		
	0	All ports	
	1-N	Specific ports	
portEn	0	Disable	1
	1	Enable	

Note *N = ptpNumberPorts*

The following table describes the portEn alternatives for portMapMode=0:

Note

startMode	*portIndex*	
	1(Slave)	*2(Master)*
4 (BC)	*0/1*	*0/1*
0,2,7 (OC-Master)	*N/A*	*0/1*
1,3,8 (OC-Slave)	*0/1*	*N/A*

The following table describes the portEn alternatives for portMapMode=>1:

Note

startMode	*portIndex*	
	Any(Slave)	*Any(Master)*
4 (BC)	*0/1*	*0/1*
0,2,7 (OC-Master)	*N/A*	*0/1*
1,3,8 (OC-Slave)	*0/1*	*N/A*

Note *The portEn control the IEEE 1588 port. It does not control the device's physical port, thus ARP and PING shall be replied by the device regardless of the portEn.*

5.3.1.31 GetIpcPtpAdminPortEnAll

Description: Get the PtpPortEnAll.

```
STATUS GetIpcPtpAdminPortEnAll(
    IN int32 fnId,
```

```
    OUT PtpPortEnAll* ptpPortEnAll);
```

Applicability	IPC9xxx
Example	rv=GetIpcPtpAdminPortEnAll(0,&ptpPortEnAll)

Parameters

Parameter Name	Range	Default
fnId	0	0
ptpPortEnAll	N/A	N/A

5.3.1.32 SetIpcPtpAdminPortTypeMode

Description: Set the portTypeMode parameter. The portTypeMode enable or disable the IEEE 1588 port groups.
type=0: All ports
type=1: Port in portState=9,8 (SLAVE, UNCALIBRATED).
type=2: Ports in portState=6,5 (MASTER, PRE-MASTER)
This API is applicable only for SetIpcPtpSdaMode(0,1).

```
STATUS SetIpcPtpAdminPortTypeMode(
    IN int32 fnId,
    IN Uint16 type,
    IN int32 portTypeMode);
```

Applicability	IPC9xxx
Example	rv=SetIpcPtpAdminPortTypeMode(0,0,0)

Parameters

Parameter Name		Range	Default
fnId	0		0
type	0	All ports	0
	1	Slave port	
	2	Master ports	
portTypeMode	0	Disable	1
	1	Enable	

The following table describes the portTypeMode alternatives:

startMode	*type*		
	1(Slave port)	*2(Master ports)*	*0(All ports)*
4 (BC)	*0/1*	*0/1*	*0/1*
0,2,7 (OC-Master)	*N/A*	*0/1*	*0/1*
1,3,8 (OC-Slave)	*0/1*	*N/A*	*0/1*

Note *The portTypeMode control the IEEE 1588 port. It does not control the device's physical port, thus ARP and PING shall be replied by the device regardless of the portEn.*

Note *Type=0 include all ports, not limited to master/slave ports.*

Note *IF (Type=0) &(portTypeMode=0) THEN Type=1,2 portTypeMode=0.*
IF (Type=0) &(portTypeMode=1) THEN Type=1,2 portTypeMode=1.

5.3.1.33 GetIpcPtpAdminPortTypeMode

Description: Get the portTypeMode parameter.

```
STATUS GetIpcPtpAdminPortTypeMode(
    IN int32 fnId,
    IN Uint16 type,
    OUT int32* portTypeMode);
```

Applicability	IPC9xxx
Example	rv=GetIpcPtpAdminPortTypeMode(0,1,&portTypeMode)

Parameters

Parameter Name	Range	Default
fnId	0	0
type	See SetIpcPtpAdminPortTypeMode	N/A
portTypeMode	See SetIpcPtpAdminPortTypeMode	N/A

5.3.1.34 GetIpcPtpAdminPortMasterMembers

Description: Get the portMembers parameter. The members are all the ports with portState=6,5 (MASTER, PRE-MASTER).

	This API is applicable only for SetIpcPtpSdaMode(0,1).
	`STATUS GetIpcPtpAdminPortMasterMembers(`
	`  IN int32 fnId,`
	`  OUT PortMembers_t* portMembers);`

Applicability	IPC9xxx
Example	rv=GetIpcPtpAdminPortMasterMembers(0,&portMembers)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	portMembers	N/A	N/A

In case of no master members:
-- terminal mode: returns portMember=0
-- host mode: returns empty (-7)

In terminal mode only valid entries shall be presented.

5.3.1.35 GetIpcPtpAdminPortSlaveMember

Description	Get the portSlave parameter. The member is the port with portState=9,8 (SLAVE, UNCALIBRATED).
	This API is applicable only for SetIpcPtpSdaMode(0,1).
	`STATUS GetIpcPtpAdminPortSlaveMember(`
	`  IN int32 fnId,`
	`  OUT Uint16* portMember);`

Applicability	IPC9xxx BC
Example	rv=GetIpcPtpAdminPortSlaveMember(0,&portMember)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	portMember	N/A	N/A

In case of no slave member:
-- terminal mode: returns portMember=0
-- host mode: returns empty (-7)

5.3.1.36 SetIpcPtpAdminMcastEthMode

Description	Set the multicast address for sending packets while in Ethernet mode. The device receive packets with both MAC addresses regardless the mcastEthMode.
	`STATUS SetIpcPtpAdminMcastEthMode(`
	`  IN int32 fnId,`
	`  IN Uint32 mcastEthMode);`

Applicability	IPC9xxx, IPC17xx
Example	rv=SetIpcPtpAdminMcastEthMode(0,0)

Parameters	Parameter Name	Range		Default
	fnId	0		0
	mcastEthMode	0	Forwardable multicast address 01-1B-19-00-00-00	0
		1	Non-forwardable multicast address 01-80-C2-00-00-0E	

Applicable for startMode = 22,23,24.

5.3.1.37 GetIpcPtpAdminMcastEthMode

Description	Get the Slave's multicast mcastEthMode mode.
	`STATUS GetIpcPtpAdminMcastEthMode(`
	`  IN int32 fnId,`
	`  OUT Uint32* mcastEthMode);`

Applicability	IPC9xxx, IPC17xx
Example	rv=GetIpcPtpAdminMcastEthMode(0,&mcastEthMode)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	mcastEthMode	See SetIpcPtpAdminMcastEthMode	0

5.3.1.38 SetIpcPtpAdminMcastIgmpMode

Description	Set the IGMP mode. This API enables or disables sending IGMP packets.
	`STATUS SetIpcPtpAdminMcastIgmpMode(`
	`  IN int32 fnId,`
	`  IN Uint32 igmpMode);`

Applicability	IPC9xxx, IPC17xx
Example	rv=SetIpcPtpAdminMcastIgmpMode(0,0)

Parameters	Parameter Name		Range	Default
	fnId	0		0
	igmpMode	0	Disable IGMP – Disable sending IGMP packets	0
		1	Enable IGMP – Enable sending IGMP packets	

In order to change the device settings, the following sequence shall be performed after invoking the API:
SetIpcAdminStore
SetIpcPtpAdminStop
SetIpcPtpAdminStart
Or
SetIpcAdminStore
SetIpcPtpAdminReset

5.3.1.39 GetIpcPtpAdminMcastIgmpMode

Description	Get the IGMP mode.
	`STATUS GetIpcPtpAdminMcastIgmpMode(`
	`  IN int32 fnId,`
	`  OUT Uint32* igmpMode);`

Applicability	IPC9xxx, IPC17xx
Example	rv=GetIpcPtpAdminMcastIgmpMode(0,&igmpMode)

Parameters	Parameter Name		Range	Default
	fnId	0		0
	igmpMode	See SetIpcPtpAdminMcastIgmpMode		

5.3.1.40 SetIpcPtpAdminMasterMode

Description	This API is applicable only for implementing boundary clock using two IPC1703 chip on FPGA, one as slave and the other as master. Setting masterMode=1 enable to set the device to be part of two chip BC. The Parent DS controlled manually by the user using SetIpcPtpAdminDsParentGm.
	`STATUS SetIpcPtpAdminMasterMode(`
	`  IN int32 fnId,`
	`  IN int32 masterMode);`

Applicability	IPC1703 – Master mode
Example	Setting Master mode to BC: rv=SetIpcPtpAdminMasterMode(0,1)

Parameters	Parameter Name		Range	Default
	fnId	0		0
	masterMode	0	Normal	0
		1	The device is part of two chips BC	

5.3.1.41 GetIpcPtpAdminMasterMode

Description	Get the Master mode of operation. This API is applicable only for implementing boundary clock using two IPC1703 chip on FPGA, one as slave and the other as master. Setting masterMode=1 enable to set the device to be part of two chip BC. The Parent DS controlled manually by the user using SetIpcPtpAdminDsParentGm.
	`STATUS GetIpcPtpAdminMasterMode(`
	`  IN int32 fnId,`
	`  OUT int32* masterMode);`

Applicability	IPC1703 – Master mode
Example	Get master mode: rv=GetIpcPtpAdminMasterMode(0,&masterMode)

Parameters	Parameter Name	Range	Default

fnId	0		0
masterMode	See	SetIpcPtpAdminMasterMode	0

5.3.1.42 SetIpcPtpAdminTxSyncPathDelay

Description	Compensate for a known delay of sync packets by adding the delay, measured in nsec.fractional_nsec, to the sync or follow_up packet's correctionField. The txPathDelay_ns is in nanoseconds while the txPathDelay_sub_ns is in 2^-16 units of nanoseconds. **STATUS SetIpcPtpAdminTxSyncPathDelay(** **IN int32 fnId,** **IN int32 pktType,** **IN int32 txPathDelay_ns,** **IN int32 txPathDelay_sub_ns);**
Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode
Example	Setting path delay to 0.5 nanoseconds in Sync packets: SetIpcPtpAdminTxSyncPathDelay(0,0,0,32768) Setting path delay to -0.5 nanoseconds in Sync packets: SetIpcPtpAdminTxSyncPathDelay(0,0,0,-32768) Setting path delay to 70000.5 nanoseconds in Follow up packets: SetIpcPtpAdminTxSyncPathDelay(0,1,70000,32768) Setting path delay to -70000.5 nanoseconds in Follow up packets: SetIpcPtpAdminTxSyncPathDelay(0,1,-70000,-32768)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	pktType	0 Sync 1 Follow Up	N/A
	txPathDelay_ns	-1000000000 – +1000000000 (+/-1 sec)	0
	txPathDelay_sub_ns	-65535 – +65535	0

It is recommended to use txPathDelay of Sync or Follow Up, but not both. In case of setting txPathDelay of the Sync packet, it is recommended to clear the txPathDelay of the Follow Up and vice versa.

5.3.1.43 GetIpcPtpAdminTxSyncPathDelay

Description	Getting the sync Tx path delay. **STATUS GetIpcPtpAdminTxSyncPathDelay(** **IN int32 fnId,** **IN int32 pktType,** **OUT TxPathDelay* txPathDelay);**
Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode
Example	rv=GetIpcPtpAdminTxSyncPathDelay(0,0,&txPathDelay)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	pktType	See SetIpcPtpAdminTxSyncPathDelay	N/A
	txPathDelay	See SetIpcPtpAdminTxSyncPathDelay	0

5.3.1.44 SetIpcPtpAdminTxDelayReqPathDelay

Description	Compensate for a known delay of delay request packets by adding the delay, measured in nsec.fractional_nsec, to the delay request correctionField. This API enables setting the asymmetry correction. The txPathDelay_ns is in nanoseconds while the txPathDelay_sub_ns is in 2^-16 units of nanoseconds. **STATUS SetIpcPtpAdminTxDelayReqPathDelay(** **IN int32 fnId,** **IN int32 pktType,** **IN int32 txPathDelay_ns,** **IN int32 txPathDelay_sub_ns);**
Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode
Example	Setting path delay to 0.5 nanoseconds: SetIpcPtpAdminTxDelayReqPathDelay(0,0,0,32768)

	Setting path delay to -0.5 nanoseconds: SetIpcPtpAdminTxDelayReqPathDelay(0,0,0,-32768) Setting path delay to 70000.5 nanoseconds: SetIpcPtpAdminTxDelayReqPathDelay(0,0,70000,32768) Setting path delay to -70000.5 nanoseconds: SetIpcPtpAdminTxDelayReqPathDelay(0,0,-70000,-32768)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	pktType	0 DelayReq	0
	txPathDelay_ns	-1000000000 – +1000000000 (+/-1 sec)	0
	txPathDelay_sub_ns	-65535 – +65535	0

5.3.1.45 GetIpcPtpAdminTxDelayReqPathDelay

Description	Getting the Delay Request Tx path delay. This API enables reading pre-set asymmetry correction.

```
STATUS GetIpcPtpAdminTxDelayReqPathDelay(
    IN int32 fnId,
    IN int32 pktType,
    OUT TxPathDelay* txPathDelay);
```

Applicability	PC9000 – BC/Slave modes, IPC17xx – Slave mode
Example	rv=GetIpcPtpAdminTxDelayReqPathDelay(0,0,&txPathDelay)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	pktType	See SetIpcPtpAdminTxDelayReqPathDelay	0
	txPathDelay	See SetIpcPtpAdminTxDelayReqPathDelay	0

5.3.1.46 SetIpcPtpAdminRxSyncPathDelay

Description	Compensate for a known delay of sync packets by subtracting the delay, measured in nsec.fractional_nsec, from the sync or follow_up packet's correctionField. This API enables setting the asymmetry correction. The rxPathDelay_ns is in nanoseconds while the rxPathDelay_sub_ns is in 2^-16 units of nanoseconds.

```
STATUS SetIpcPtpAdminRxSyncPathDelay(
    IN int32 fnId,
    IN int32 pktType,
    IN int32 rxPathDelay_ns,
    IN int32 rxPathDelay_sub_ns);
```

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode
Example	Setting path delay to 0.5 nanoseconds in Sync packets: SetIpcPtpAdminRxSyncPathDelay(0,0,0,32768) Setting path delay to -0.5 nanoseconds in Sync packets: SetIpcPtpAdminRxSyncPathDelay(0,0,0,-32768) Setting path delay to 70000.5 nanoseconds in Follow up packets: SetIpcPtpAdminRxSyncPathDelay(0,1,70000,32768) Setting path delay to -70000.5 nanoseconds in Follow up packets: SetIpcPtpAdminRxSyncPathDelay(0,1,-70000,-32768)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	pktType	0 Sync 1 Follow Up	N/A
	rxPathDelay_ns	-1000000000 – +1000000000 (+/-1 sec)	0
	rxPathDelay_sub_ns	-65535 – +65535	0

5.3.1.47 GetIpcPtpAdminRxSyncPathDelay

Description	Getting the sync Rx path delay. This API enables reading pre-set asymmetry correction.

```
STATUS GetIpcPtpAdminRxSyncPathDelay(
    IN int32 fnId,
    IN int32 pktType,
```

	OUT RxPathDelay* rxPathDelay);
Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode
Example	rv=GetIpcPtpAdminRxSyncPathDelay(0,0,&rxPathDelay)

Parameters

Parameter Name		Range	Default
fnId	0		0
pktType	See	SetIpcPtpAdminRxSyncPathDelay	N/A
rxPathDelay	See	SetIpcPtpAdminRxSyncPathDelay	0

5.3.1.48 SetIpcPtpAdminRxSyncRate

Description: Set the slave's expected Sync packet rate.

```
STATUS SetIpcPtpAdminRxSyncRate(
    IN int32 fnId,
    IN Uint32 slavePktRate);
```

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=SetIpcPtpAdminRxSyncRate(0,16)

Parameters

Parameter Name	Range	Default
fnId	0	0
slavePktRate	2^N (N=1…7)	32 (profile-0,1) 16 (profile-2)

(1) This API is applicable only if FLASH memory and configDB are in use.
(2) The new Sync packet rate will take effect only after storing configDB in FLASH memory and performing Stop/Start.

5.3.1.49 GetIpcPtpAdminRxSyncRate

Description: Get the slave's expected Sync packet rate.

```
STATUS GetIpcPtpAdminRxSyncRate(
    IN int32 fnId,
    OUT Uint32* slavePktRate);
```

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=GetIpcPtpAdminRxSyncRate(0,&slavePktRate)

Parameters

Parameter Name	Range	Default
fnId	0	0
slavePktRate	See SetIpcPtpAdminRxSyncRate	

5.3.1.50 SetIpcPtpAdminDelayReqInterval

Description: Set the delay request interval (ptpDelayReqInterval) as follow:
ptpDelayReqInterval = pktRate/delayReqRate
The ptpDelayReqInterval defined as the ratio between the sync packet rate as been set by pktRate of SetIpcPtpAdminStart (or slavePktRate provided by GetIpcPtpAdminRxSyncRate) and the delayReqRate.
The ptpDelayReqInterval can be set in 2^N steps where 2^N = 1, 2, 4, 8, 16, … up to 2^(4+log2(pktRate)).
As examples:
(1) pktRate=32pkt/sec, delayReqRate=8pkt/sec – set ptpDelayReqInterval=4
(2) pktRate=32pkt/sec, delayReqRate=1/16pkt/sec – set ptpDelayReqInterval=512

```
STATUS SetIpcPtpAdminDelayReqInterval(
    IN int32 fnId,
    IN Uint32 ptpDelayReqInterval);
```

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=SetIpcPtpAdminDelayReqInterval(0,4)

Parameters

Parameter Name	Range	Default
fnId	0	0
ptpDelayReqInterval	1 .. 2^(4+log2(pktRate))	4 (profile-0,1) 1 (profile-2)

Note *For high performance, the recommended DelayReq packet rate range is 8-32pkt/sec.*

Note *logMinDelayReqInterval = - LOG2 (GetIpcPtpAdminRxSyncRate / GetIpcPtpAdminDelayReqInterval)*

Note *In profile 2, SetIpcPtpAdminDelayReqInterval doesn't stored to ConfDB.*

5.3.1.51 GetIpcPtpAdminDelayReqInterval

Description	Get the delay request interval (ptpDelayReqInterval). The ptpDelayReqInterval defines as the ratio between the pktRate and the delayReqRate.

```
STATUS GetIpcPtpAdminDelayReqInterval(
    IN int32 fnId,
    OUT Uint32* ptpDelayReqInterval);
```

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603		
Example	rv=GetIpcPtpAdminDelayReqInterval(0,&ptpDelayReqInterval)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	ptpDelayReqInterval	See SetIpcPtpAdminDelayReqInterval	

5.3.1.52 SetIpcPtpAdminTwoStep

Description	Set the Slave's Two Step mode. When set to 0, the Slave ignores follow up packets even if the twoStepFlag bit is set in the Sync packet flagField. When set to 1, the Slave acts in accordance with the twoStepFlag value in the Sync packet flag field.

```
STATUS SetIpcPtpAdminTwoStep(
    IN int32 fnId,
    IN int32 slaveTwoStepMode);
```

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603			
Example	rv=SetIpcPtpAdminTwoStep(0,0)			
Parameters	Parameter Name		Range	Default
	fnId	0		0
	slaveTwoStepMode	0	Enables one step only	1
		1	Enables one step or two step	

5.3.1.53 GetIpcPtpAdminTwoStep

Description	Get the Slave's Two Step mode.

```
STATUS GetIpcPtpAdminTwoStep(
    IN int32 fnId,
    OUT int32* slaveTwoStepMode);
```

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=GetIpcPtpAdminTwoStep(0,&slaveTwoStepMode)

5.3.1.54 SetIpcPtpAdminCorrectionFieldEn

Description	Set the Slave's correction field handling operational mode. When enabled, the Slave add the correction field value to the algorithm suite.

```
STATUS SetIpcPtpAdminCorrectionFieldEn(
    IN int32 fnId,
    IN int32 slaveCorrectionFieldEn);
```

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603			
Example	rv=SetIpcPtpAdminCorrectionFieldEn(0,1)			
Parameters	Parameter Name		Range	Default
	fnId	0		0
	slaveCorrectionFieldEn	0	Disable using correction field	1
		1	Enable using correction field	

5.3.1.55 GetIpcPtpAdminCorrectionFieldEn

Description	Get the Slave's correction field handling operational mode.

```
STATUS GetIpcPtpAdminCorrectionFieldEn(
    IN int32 fnId,
    OUT int32* slaveCorrectionFieldEn);
```

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=GetIpcPtpAdminCorrectionFieldEn(0,&slaveCorrectionFieldEn)

Parameters	Parameter Name		Range	Default
	fnId	0		0
	slaveCorrectionFieldEn	See	SetIpcPtpAdminCorrectionFieldEn	1

5.3.1.56 SetIpcPtpAdminCorrectionFieldReportEn

Description	Set the Slave's correction field report mode. When enabled, the Slave print the correction field statistics.

```
STATUS SetIpcPtpAdminCorrectionFieldReportEn(
    IN int32 fnId,
    IN int32 slaveCorrectionFieldReportEn);
```

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=SetIpcPtpAdminCorrectionFieldReportEn(0,1)

Parameters	Parameter Name		Range	Default
	fnId	0		0
	slaveCorrectionFieldReportEn	0	Disable Report	0
		1	Enable Report	

5.3.1.57 GetIpcPtpAdminCorrectionFieldReportEn

Description	Get the Slave's correction field report mode. When enabled, the Slave print the correction field statistics.

```
STATUS GetIpcPtpAdminCorrectionFieldReportEn(
    IN int32 fnId,
    OUT int32* slaveCorrectionFieldReportEn);
```

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=GetIpcPtpAdminCorrectionFieldReportEn(0,&slaveCorrectionFieldReportEn)

Parameters	Parameter Name		Range	Default
	fnId	0		0
	slaveCorrectionFieldReportEn	See	SetIpcPtpAdminCorrectionFieldReportEn	

5.3.1.58 SetIpcPtpAdminCorrectionFieldLimitMode

Description	Set the Slave's correction field limit mode.

```
STATUS SetIpcPtpAdminCorrectionFieldLimitMode(
    IN int32 fnId,
    IN int32 slaveCFLimitMode);
```

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=SetIpcPtpAdminCorrectionFieldLimitMode(0,1)

Parameters	Parameter Name		Range	Default
	fnId	0		0
	slaveCFLimitMode	0	Without limit	1
		1	Limit-A. If limit exceeded, the CF is replaced by previous CF.	
		2	Limit-B. If limit exceeded, CF=0.	

5.3.1.59 GetIpcPtpAdminCorrectionFieldLimitMode

Description	Get the Slave's correction field limit mode.

```
STATUS GetIpcPtpAdminCorrectionFieldLimitMode(
    IN int32 fnId,
    OUT int32* slaveCFLimitMode);
```

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=GetIpcPtpAdminCorrectionFieldLimitMode(0,&slaveCFLimitMode)

Parameters	Parameter Name		Range	Default
	fnId	0		0
	slaveCFLimitMode	See	SetIpcPtpAdminCorrectionFieldLimitMode	1

5.3.1.60 SetIpcPtpAdminCorrectionFieldLimitTh

Description	Set the Slave's correction field limit threshold in nanoseconds.

```
STATUS SetIpcPtpAdminCorrectionFieldLimitTh(
```

	IN int32 fnId, **IN int32 slaveCFLimitTh);**
Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=SetIpcPtpAdminCorrectionFieldLimitTh(0,10000)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	slaveCFLimitTh	10 - 100000000	50000000

5.3.1.61 GetIpcPtpAdminCorrectionFieldLimitTh

Description	Get the Slave's correction field limit threshold in nanoseconds.
	STATUS GetIpcPtpAdminCorrectionFieldLimitTh(**IN int32 fnId,** **OUT int32* slaveCFLimitTh);**
Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=GetIpcPtpAdminCorrectionFieldLimitTh(0,&slaveCFLimitTh)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	slaveCFLimitTh	See SetIpcPtpAdminCorrectionFieldLimitTh	50000000

5.3.2 Hybrid IEEE 1588 & SyncE Operation

The IPC1603, IPC17xx and IPC9xxx support hybrid IEEE 1588 & SyncE operation in conjunction with external 3rd party PLL. The specific hybrid operation and features depend on the specific product:

- IPC1603/IPC17xx/IPC9xxx – Hybrid by operation in conjunction with external 3rd party PLL (no preference). SyncE recovered clock shall be connected to the OSCIN pin. The clock injected to OSCIN shall be valid continuously after device reset.
- IPC9000/IPC9100/IPC17xx – Hybrid by operation in conjunction with external 3rd party PLL (no preference). SyncE recovered clock shall be connected to the SYSIN or CLKIN pins. The device automatically detects and uses the clock at SYSIN or CLKIN pins.
- IPC9600 - Hybrid by operation in conjunction with external ADI AD9548 PLL. The IPC9600 and AD9548 operate as a fully integrate IEEE 1588 & syncE chipset.

The IPC9xxx and IPC17xx device enables several pin configurations hybrid clock modes (ClkMode) controlled by SetIpcPtpAdminHybridClkMode API:

- ClkMode = 0 - Device uses OSCIN pin as reference clock and for device operation. The clock provided to OSCIN pin shall be IPClock approved (or better) oscillator or the output of low-jitter network (WAN) PLL (e.g. EEC-PLL).
- ClkMode = 1 - Device uses 10MHz SYSIN or CLKIN pins, if valid clocks, as reference clock, and OSCIN pin for device operation. In case SYSIN or CLKIN clock is not valid, the device uses OSCIN as reference clock as well. The clock provided to SYSIN or CLKIN pins shall be the output of low-jitter network (WAN) PLL (e.g. EEC-PLL) or IPClock approved (or better) oscillator. The user sets the device to use SYSIN pin or CLKIN pin as the hybrid clock source by using SetIpcAdminClkSysInMuxSel API.

In all modes, the device uses clock provided to OSCIN pin for operation. OSCIN clock must be valid under all conditions.

The IPC1703 and IPC1603 devices support ClkMode=0 only. The IPC17xx device supports ClkMode=0,1.

The signal at SYSIN-CLKIN, better be valid before the device is been operated. In case signal was not valid, the device will select OSCIN. In case the signal at SYSIN-CLKIN was valid and was selected, and later it became not valid, the device will automatically switch to OSCIN. This switching from SYSIN-CLKIN frequency to OSCIN frequency may force the device to relock. Such a relock will typically accomplished within 10-15 minutes. However it may take longer. Similar pattern will happen in case SYSIN-CLKIN was not valid and later became valid. The term "not valid" means that there is no clock signal at all.

Table 9 summarized the ClkMode operation modes:

Clk Mode	**ClkMode Description**	**OSCIN**	**SYSIN-CLKIN**
0	OSCIN	20MHz or 12.8MHz or 10MHz from oscillator	N/A
1	Auto SYSIN-CLKIN / OSCIN		10MHz from EEC-PLL

Table 9: ClkMode Operation

The device's servo supports hybrid IEEE 1588 and SyncE operation. While using ClkMode=1,2, in order to enable hybrid operation, the recovered SyncE clock shall be provided by external PLL connected to the device's OSCIN / SYSIN-CLKIN pin. The user can select one of the following clock recovery modes (slaveHybridMode) of operation using SetIpcPtpAdminHybridMode API:

- slaveHybridMode = 0 - IEEE 1588 & SyncE. Frequency is recovered by IEEE 1588 and optionally by SyncE. In this mode the SyncE clock recovery (done by the external PLL) is not mandatory. Time is recovered by IEEE 1588.
- slaveHybridMode = 1 - SyncE only. Frequency is recovered by SyncE only. In this mode, the SyncE clock recovery (done by the external PLL) is mandatory. Time is recovered by IEEE 1588.

It is recommended to set the slaveHybridMode before the slave port start to lock on the master port. Changes of the settings of the slaveHybridMode usually enforced within up to 5 minutes.

Hybrid operation of IEEE 1588 and SyncE shall improve the synchronization performance, however in specific conditions the performance may be severely degraded. The term synchronization performance refers to lock time, time error, fractional frequency offset and HO frequency accuracy.
Table 10 summarized the sync performance level in case of clkMode=1,2 for various hybrid setups. For simplicity, 4 synchronization levels will be used: Level-1 to 4. Level-1 is the lowest grade while Level-4 is the highest grade. Level-2 is the normal grade delivered while using IEEE 1588 only, without SyncE.
Note that in order to avoid Level-1 while using clkMode=1,2, slaveHybridMode shall be set to 1 only in case of valid SyncE available at SYSIN-CLKIN can be ensured.

Synchronization Performance	**Valid SyncE clock available at SYSIN-CLKIN**	**clkMode**	**slaveHybridMode**	**Comment**
Level-4	Yes	1	1	Best
Level-3	Yes	1	0	Improved
Level-2	No	1	0	Normal
Level-1	No	1	1	Severely Degraded

Table 10: Synchronization Performance for Various Hybrid Setups

5.3.2.1 SetIpcPtpAdminHybridClkMode

Description: Set the hybrid clock mode (ClkMode). The hybrid clock mode controls the device operation using OSCIN, SYSIN (or CLKIN) pins.

```
STATUS SetIpcPtpAdminHybridClkMode(
    IN int32 fnId,
    IN int32 ClkMode);
```

Applicability: IPC9xxx, IPC17xx

Example: rv=SetIpcPtpAdminHybridClkMode(0,1)

Parameters:

Parameter Name		Range	Default
fnId	0		0
ClkMode	0	OSCIN	0
	1	Auto – OSCIN or SYSIN/CLKIN	

In order to change the device settings, the following sequence shall be performed after invoking the API:
SetIpcAdminStore
SetIpcPtpAdminStop
SetIpcPtpAdminStart
Or
SetIpcAdminStore
SetIpcPtpAdminReset

In ClkMode=1 the device automatically select the OSCIN or SYSIN/CLKIN. The user shall select SYSIN or CLKIN as the source for the hybrid operation using SetIpcAdminClkSysInMuxSel.

ClkMode=2 supported in IPC9xxx only.

5.3.2.2 GetIpcPtpAdminHybridClkMode

Description	Get the hybrid clock mode (ClkMode).
	`STATUS GetIpcPtpAdminHybridClkMode(`
	`  IN int32 fnId,`
	`  OUT int32* ClkMode);`

Applicability	IPC9xxx, IPC17xx
Example	rv=GetIpcPtpAdminHybridClkMode(0,&ClkMode)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	ClkMode	See SetIpcPtpAdminHybridClkMode	0

5.3.2.3 GetIpcPtpAdminHybridClkStatus

Description	Get the hybrid clock status (hybridClkStatus).
	`STATUS GetIpcPtpAdminHybridClkStatus(`
	`  IN int32 fnId,`
	`  OUT int32* hybridClkStatus);`

Applicability	IPC9xxx, IPC17xx
Example	rv=GetIpcPtpAdminHybridClkMode(0,&hybridClkStatus)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	hybridClkStatus	0 - OSCIN	
		1 - SYSIN	

5.3.2.4 SetIpcPtpAdminHybridMode

Description	Set the Slave's hybrid mode of operation.
	`STATUS SetIpcPtpAdminHybridMode(`
	`  IN int32 fnId,`
	`  IN int32 slaveHybridMode);`

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=SetIpcPtpAdminHybridMode(0,1)

Parameters	Parameter Name	Range		Default
	fnId	0		0
	slaveHybridMode	0	IEEE 1588 & SyncE	0
		1	SyncE only	

5.3.2.5 GetIpcPtpAdminHybridMode

Description	Get the Slave's hybrid mode of operation.
	`STATUS GetIpcPtpAdminHybridMode(`
	`  IN int32 fnId,`
	`  OUT int32* slaveHybridMode);`

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=GetIpcPtpAdminHybridMode(0,&slaveHybridMode)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	slaveHybridMode	See SetIpcPtpAdminHybridMode	0

5.3.2.6 SetIpcPtpAdminHybridSyncEClkMode

Description	Set the hybrid SyncE clock mode (HybridSyncEClkMode). The HybridSyncEClkMode controls the operation with Ethernet PHY in order to support driving or using SyncE clock.
	`STATUS SetIpcPtpAdminHybridSyncEClkMode(`
	`  IN int32 fnId,`
	`  IN int32 HybridSyncEClkMode,`
	`  IN int32 Reserved);`

Applicability	IPC9600
Example	rv=SetIpcPtpAdminHybridSyncEClkMode(0,2,0)

Parameters	Parameter Name	Range	Default
	fnId	0	0

HybridSyncEClkMode	0	Disable	0
	1	SyncE master	
	2	SyncE slave - extDevMode=43	
	3	Reserved	
	4	SyncE slave - extDevMode=45, 47	
	10	Idle – Do not change current settings	
Reserved	0		0

5.3.2.7 GetIpcPtpAdminHybridSyncEClkMode

Description	Get the hybrid SyncE clock mode (HybridSyncEClkMode).
	`STATUS GetIpcPtpAdminHybridSyncEClkMode(`
	`  IN int32 fnId,`
	`  OUT int32* HybridSyncEClkMode);`

Applicability	IPC9600
Example	rv=GetIpcPtpAdminHybridSyncEClkMode(0,&HybridSyncEClkMode)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	HybridSyncEClkMode	See SetIpcPtpAdminHybridSyncEClkMode	0

5.3.3 Time

5.3.3.1 Common

5.3.3.1.1 GetIpcPtpTimePtpTime

Description	Get the current time in PTP format (seconds).
	In order to avoid ambiguity, this API returns the current time only when it is been called in the time period starting approximately 100ms after the rising edge of the 1PPS, ending at the next 1PPS rising edge. In case this API is been used during approximately the first 100ms after the rising edge of the 1PPS, the API returns error and does not return the time. As a result, in case of none synchronized use of this API, about 10% of the calls will not return the time.
	GetIpcPtpTimePtpTimeNs returns the current time regardless of the 1PPS.
	`STATUS GetIpcPtpTimePtpTime(`
	`  IN int32 fnId,`
	`  OUT IPC_Uint48* ptpTime);`

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=GetIpcPtpTimePtpTime(0,&ptpTime)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	ptpTime	N/A	N/A

5.3.3.1.2 GetIpcPtpTimePtpTimeNs

Description	Get the current time in PTP format includes nanoseconds (seconds:nsec).
	`STATUS GetIpcPtpTimePtpTimeNs(`
	`  IN int32 fnId,`
	`  OUT PtpTimeNs* ptpTimeNs);`

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=GetIpcPtpTimePtpTimeNs(0,&ptpTimeNs)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	ptpTimeNs	N/A	N/A

5.3.3.1.3 GetIpcPtpTimeGpsTime

Description	Get the current time in GPS format (weeks:seconds). The number of weeks returned assumes rollover = 1 hence to get GPS weeks with no rollover requires adding 1024 to gpsTime.gpsWeeks. In order to avoid ambiguity, this API returns the current time only when it is been called in the time period starting approximately 100ms after the rising edge of the

	1PPS, ending at the next 1PPS rising edge. In case this API is been used during approximately the first 100ms after the rising edge of the 1PPS, the API returns error and does not return the time. As a result, in case of none synchronized use of this API, about 10% of the calls will not return the time. GetIpcPtpTimeGpsTimeNs returns the current time regardless of the 1PPS. **STATUS GetIpcPtpTimeGpsTime(** **IN int32 fnId,** **OUT GpsTime* gpsTime);**
Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=GetIpcPtpTimeGpsTime(0,&gpsTime)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	gpsTime	N/A	N/A

5.3.3.1.4 GetIpcPtpTimeGpsTimeNs

Description	Get the current time in GPS format includes nanoseconds (weeks:seconds:nsec). **STATUS GetIpcPtpTimeGpsTimeNs(** **IN int32 fnId,** **OUT GpsTmeNs* gpsTimeNs);**
Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=GetIpcPtpTimeGpsTimeNs(0,&gpsTimeNs)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	gpsTimeNs	N/A	N/A

5.3.3.1.5 GetIpcPtpTimeUtcTime

Description	Get the current time in UTC format (YYYY-MM-DD hh:mm:dd). In order to avoid ambiguity, this API returns the current time only when it is been called in the time period starting approximately 100ms after the rising edge of the 1PPS, ending at the next 1PPS rising edge. In case this API is been used during approximately the first 100ms after the rising edge of the 1PPS, the API returns error and does not return the time. As a result, in case of none synchronized use of this API, about 10% of the calls will not return the time. GetIpcPtpTimeUtcTimeNs returns the current time regardless of the 1PPS. **STATUS GetIpcPtpTimeUtcTime(** **IN int32 fnId,** **OUT UtcTime* utcTime);**
Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=GetIpcPtpTimeUtcTime(0,&utcTime)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	utcTime	N/A	N/A

5.3.3.1.6 GetIpcPtpTimeUtcTimeNs

Description	Get the current time in UTC format includes nanoseconds (YYYY-MM-DD hh:mm:dd:nsec). **STATUS GetIpcPtpTimeUtcTimeNs(** **IN int32 fnId,** **OUT UtcTimeNs* utcTimeNs);**
Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=GetIpcPtpTimeUtcTimeNs(0,&utcTimeNs)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	utcTimeNs	N/A	N/A

5.3.3.1.7 SetIpcPtpTimeMode

Description	Set the ptpTimeMode. The ptpTimeMode enables to use the stored time, as was stored in the configuration database (configDB) on the local FLASH memory, and to use it during the next power cycle or reset. The user shall invoke

	SetIpcAdminStore(0,0) in order to perform the store to FLASH memory.
	`STATUS SetIpcPtpTimeMode(`
	`  IN int32 fnId,`
	`  IN int32 ptpTimeMode);`

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=SetIpcPtpTimeMode(0,1)

Parameters	Parameter Name	Range		Default
	fnId	0		0
	ptpTimeMode	0	Do not use the stored time	0
		1	Use the stored time	

5.3.3.1.8 GetIpcPtpTimeMode

Description	Get the ptpTimeMode.
	`STATUS GetIpcPtpTimeMode(`
	`  IN int32 fnId,`
	`  OUT int32* ptpTimeMode);`

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=GetIpcPtpTimeMode(0,&ptpTimeMode)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	ptpTimeMode	See SetIpcPtpTimeMode	0

5.3.3.2 Master

The time of the IPC9xxx or IPC17xx Master mode can be set using one of the following alternatives:

1. TOD NMEA interface – as describes in datasheet, section Time of Day Interface
2. API – SetIpcPtpTimePtpTime
3. API – SetIpcPtpTimeGpsTime
4. API – SetIpcPtpTimeGpsTime1

5.3.3.2.1 SetIpcPtpTimePtpTime

Description	Set the Master's current time in PTP format (seconds).
	This API returns the current time only when it been called 100ms after the rising edge of the 1PPS. In case of none synchronized use of the API, about 10% of the calls will not return the time.
	`STATUS SetIpcPtpTimePtpTime(`
	`  IN int32 fnId,`
	`  IN Uint16 ptpTimeHigh,`
	`  IN Uint32 ptpTimeLow);`

Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode
Example	rv=SetIpcPtpTimePtpTime(0,0,1211367395)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	ptpTimeHigh	0 - 65536	N/A
	ptpTimeLow	0 - 4294967296	N/A

Can be used also in IPC9xxx – BC mode while Slave is in free run or holdover state.

5.3.3.2.2 SetIpcPtpTimeGpsTime

Description	Sets the current Master's time in GPS format (weeks:seconds).
	This API function should be called right after the 1PPS signal rising edge.
	The number of weeks expected assumes rollover = 1 hence GPS weeks with no rollover requires subtracting 1024 before assigning it to gpsWeeks.
	This API returns the current time only when it been called 100ms after the rising edge of the 1PPS. In case of none synchronized use of the API, about 10% of the calls will not return the time.
	`STATUS SetIpcPtpTimeGpsTime(`
	`  IN int32 fnId,`
	`  IN Uint32 gpsWeeks,`

	IN Uint32 secInLastWeek);
Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode
Example	rv=SetIpcPtpTimeGpsTime(0,634,172355)

Parameters	Parameter Name	Range	Default
	fnId	0	N/A
	gpsWeeks	20 - 1024	N/A
	secInLastWeek	0 - 604800	N/A

Can be used also in IPC9xxx – BC mode while Slave is in free run or holdover state.

The supported UTC time starting at 2000-01-01.

5.3.3.2.3 SetIpcPtpTimeGpsTime1

Description	Similar to SetIpcPtpTimeGpsTime however in addition the API causes master to move into trace status. Sets the current Master's time in GPS format (weeks:seconds). This API function should be called right after the 1PPS signal rising edge. The number of weeks expected assumes rollover = 1 hence GPS weeks with no rollover requires subtracting 1024 before assigning it to gpsWeeks. This API returns the current time only when it been called 100ms after the rising edge of the 1PPS. In case of none synchronized use of the API, about 10% of the calls will not return the time. **STATUS SetIpcPtpTimeGpsTime(** **IN int32 fnId,** **IN Uint32 gpsWeeks,** **IN Uint32 secInLastWeek);**
Applicability	IPC9xxx – Master mode, IPC17xx – Master mode
Example	rv=SetIpcPtpTimeGpsTime1(0,634,172355)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	gpsWeeks	20 - 1024	N/A
	secInLastWeek	0 - 604800	N/A

SetIpcPtpTimeGpsTime1 cause master to move into trace status as can be observed by GetIpcPtpState.

The supported UTC time starting at 2000-01-01.

5.3.4 Default Dataset

5.3.4.1 GetIpcPtpDsDefault

Description	Get default DS. **STATUS GetIpcPtpDsDefault(** **IN int32 fnId,** **OUT DefaultDs_t* defaultDs);**
Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=GetIpcPtpDsDefault(0,&defaultDs)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	defaultDs	N/A	N/A

5.3.4.1.1 Default Values

The default parameters in defaultDs are provided below.

Parameter	Master	Slave	BC
Two-step flag	0 (1 for IPC1703)	0	0
Clock Identity	00:0A:35:FF:FE:01:F4:15	00:0A:35:FF:FE:01:F4:14	00:0A:35:FF:FE:01:F4:16
Number ports	1	1	2

Clock Quality:			
ClockClass	248	255	248
ClockAccuracy	254 (profile-0,1,2)	254 (profile-0,1,2)	254 (profile-0,1,2)
OffsetScaledLogVar	65535 (profile-0,1,2)	65535 (profile-0,1,2)	65535 (profile-0,1,2)
Priority1	128	255	128
Priority2	128	255	128
Domain number	0 (profile-0) 4 (profile-1) 24 (profile-2)	0 (profile-0) 4 (profile-1) 24 (profile-2)	0 (profile-0) 4 (profile-1) 24 (profile-2)
Slave only	0	1	0
LocalPriority	128	128	128

5.3.4.2 SetIpcPtpDsTwoStepFlag

Description	Set the defaultDs.twoStepFlag parameter. This set for sending Follow Up messages.
	`STATUS SetIpcPtpDsTwoStepFlag(`
	`  IN int32 fnId,`
	`  IN int32 ptpTwoStepFlag);`

Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode
Example	Enable Two Step: rv=SetIpcPtpDsTwoStepFlag(0,1)

Parameters

Parameter Name		Range	Default
fnId	0		0
ptpTwoStepFlag	0	One Step	IPC9xxx – 0
	1	Two Step	IPC17xx – 0 IPC1703 – 1

Note

One Step – supported by IPC9xxx and IPC17xx
Two Steps – supported by IPC9xxx, IPC17xx

5.3.4.3 GetIpcPtpDsTwoStepFlag

Description	Get the defaultDs.twoStepFlag parameter.
	`STATUS GetIpcPtpDsTwoStepFlag(`
	`  IN int32 fnId,`
	`  OUT int32* ptpTwoStepFlag);`

Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode
Example	rv=GetIpcPtpDsTwoStepFlag(0,&ptpTwoStepFlag)

Parameters

Parameter Name	Range	Default
fnId	0	0
ptpTwoStepFlag	See SetIpcPtpDsTwoStepFlag	

5.3.4.4 SetIpcPtpDsClockIdentity

Description	Set the defaultDs.clockIdentity parameter. The value of defaultDs.clockIdentity is placed in all IEEE 1588 packets header portIdentity.clockIdentity field.
	`STATUS SetIpcPtpDsClockIdentity(`
	`  IN int32 fnId,`
	`  IN char* ptpClockIdentity);`

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=SetIpcPtpDsClockIdentity(0,“00:0A:35:FF:FE:01:F4:16”)

Parameters

Parameter Name	Range	Default
fnId	0	0
ptpClockIdentity	N/A	“00:0A:35:FF:FE:01:F4:15” – Master “00:0A:35:FF:FE:01:F4:14” – Slave “00:0A:35:FF:FE:01:F4:16” – BC

5.3.4.5 GetIpcPtpDsClockIdentity

Description	Get the defaultDs.clockIdentity parameter.
	`STATUS GetIpcPtpDsClockIdentity(`
	`  IN int32 fnId,`
	`  OUT char** ptpClockIdentity);`

Applicability	IPC9xxx, IPC17xx, IPC1603

Example	rv=GetIpcPtpDsClockIdentity(0,&ptpClockIdentity)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	ptpClockIdentity	See SetIpcPtpDsClockIdentity	

5.3.4.6 SetIpcPtpDsNumPorts

Description	Set the defaultDs.numberPorts parameter. **STATUS SetIpcPtpDsNumPorts(** **IN int32 fnId,** **IN Uint16 ptpNumberPorts);**		
Applicability	IPC9xxx, IPC17xx		
Example	rv=SetIpcPtpDsNumPorts(0,2)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	ptpNumberPorts	portMapMode=0: Slave/Master – 1 BC – 2..64 portMapMode=>1: Slave/Master – 1..64 BC – 2..64	portMapMode=0: Slave/Master – 1 BC – 2 portMapMode=>1: Slave/Master – 8 BC – 8

In order to change the device settings, the following sequence shall be performed after invoking the API:
SetIpcAdminStore
SetIpcPtpAdminStop
SetIpcPtpAdminStart
Or
SetIpcAdminStore
SetIpcPtpAdminReset

5.3.4.7 GetIpcPtpDsNumPorts

Description	Get the defaultDs.numberPorts parameter. **STATUS GetIpcPtpDsNumPorts(** **IN int32 fnId,** **OUT Uint16* ptpNumberPorts);**		
Applicability	IPC9xxx, IPC17xx		
Example	rv=GetIpcPtpDsNumPorts(0,&ptpNumberPorts)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	ptpNumberPorts	See SetIpcPtpDsNumPorts	

5.3.4.8 SetIpcPtpDsClockClass

Description	Set the defaultDs.clockQuality.clockClass parameter. **STATUS SetIpcPtpDsClockClass(** **IN int32 fnId,** **IN u_char ptpClockClass);**				
Applicability	IPC9xxx, IPC17xx, IPC1603				
Example	rv=SetIpcPtpDsClockClass(0,7)				
Parameters	Parameter Name	Range		Default	
	fnId	0		0	
	ptpClockClass	6	A clock that is synchronized to a primary reference time source with PTP timescale	255 248 248	Slave Master BC
		7	A clock that has previously been designated as clockClass 6 and is in holdover mode and within holdover specifications [T-GM]		
		13	A clock that is synchronized to a primary reference time source with ARB timescale		
		14	A clock that has previously been		

	designated as clockClass 13 and is in holdover mode and within holdover specifications
52	A clock that has previously been designated as clockClass 7 and is in holdover mode and outside holdover specifications
58	A clock that has previously been designated as clockClass 14 and is in holdover mode and outside holdover specifications
102	Free run – Stratum 3
135	T-BC in holdover, within holdover specification
140	T-GM in holdover, out of holdover specification, traceable to Category 1 frequency source
150	T-GM in holdover, out of holdover specification, traceable to Category 2 frequency source
160	T-GM in holdover, out of holdover specification, traceable to Category 3 frequency source
165	T-BC in holdover, out of holdover specification, using unspecified frequency source
248	Unknown – was never locked T-GM or T-BC without time reference since start-up
255	Slave only

Note *clockClass can be set to any valid value in accordance with IEEE 1588 profiles.*

Note *In case of master modes, while SetIpcAdminClkInRefMode in Auto mode: clockClass=6 in Lock, clockClass=7 in Holdover, and clockClass=52 in Trace.*

5.3.4.9 GetIpcPtpDsClockClass

Description	Get the defaultDs.clockQuality.clockClass parameter.

```
STATUS GetIpcPtpDsClockClass(
    IN int32 fnId,
    OUT u_char* ptpClockClass);
```

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=GetIpcPtpDsClockClass(0,&ptpClockClass)

Parameters

Parameter Name	Range	Default
fnId	0	0
ptpClockClass	See SetIpcPtpDsClockClass	

5.3.4.10 SetIpcPtpDsClockClassPar

Description	Set the clockClassPar.

```
STATUS SetIpcPtpDsClockClassPar(
    IN  int32 fnId,
    IN  u_char DefCcTr,
    IN  u_char DefCcLK);
```

Applicability	IPC9xxx – BC
Example	rv=SetIpcPtpDsClockClassPar(0,248,248)

Parameters

Parameter Name	Range	Default
fnId	0	0
DefCcTr	0 - 255	248
DefCcLk	0 - 255	248

5.3.4.11 GetIpcPtpDsClockClassPar

Description	Get the clockClassPar.
	`STATUS GetIpcPtpDsClockClassPar(`
	`  IN  int32 fnId,`
	`  OUT clockClassPar* clockClassPar);`

Applicability	IPC9xxx – BC
Example	rv=GetIpcPtpDsClockClassPar(0,&clockClassPar)

Parameters

Parameter Name	Range	Default
fnId	0	0
clockClassPar	See SetIpcPtpDsClockClassPar	N/A

5.3.4.12 SetIpcPtpDsDefaultMode

Description: Set the defaultDs mode of operation while in BC mode.

Manual-DefDs	The defaultDs.clockQuality.clockClass set manually using the SetIpcPtpDsClockClass API.
Auto	The defaultDs.clockQuality.clockClass set automatically.
Auto-A	The defaultDs.clockQuality.clockClass set automatically using DefCcTr and DefCcLk as been set by SetIpcPtpDsClockClassPar.

```
STATUS SetIpcPtpDsDefaultMode(
    IN int32 fnId,
    IN int32 dsDefMode);
```

Applicability: IPC9xxx BC mode

Example: rv=SetIpcPtpDsDefaultMode(0,6)

Parameters

Parameter Name	Range	Default
fnId	0	0
dsDefMode	4 – Manual-DefDs 6 – Auto 7 – Auto-A	6

Note

ClockClass for IPC9xxx-BC mode, dsDefMode=6,7:

Description	*Gstate.state*	*HO Spec*	*ClockClass*		
			startMode=4	*startMode=24*	
			dsDefMode=6,7	*dsDefMode=6*	*dsDefMode=7*
FR	*1*	*N/A*	*248 (def)*	*248 (def)*	*248 (def)*
HO	*2*	*Out of spec*	*248 (def)*	*165*	*165*
HO	*2*	*In spec*	*248 (def)*	*135*	*135*
TR	*3*	*N/A*	*248 (def)*	*248 (def)*	*248 (DefCcTr)*
LK	*4*	*N/A*	*248 (def)*	*248 (def)*	*248 (DefCcLk)*

5.3.4.13 GetIpcPtpDsDefaultMode

Description	Get the defaultDs mode of operation while in BC mode.
	`STATUS GetIpcPtpDsDefaultMode(`
	`  IN int32 fnId,`
	`  OUT int32* dsDefMode);`

Applicability	IPC9xxx BC mode
Example	rv=GetIpcPtpDsDefaultMode(0,&dsDefMode)

Parameters

Parameter Name	Range	Default
fnId	0	0
dsDefMode	See SetIpcPtpDsDefaultMode	

5.3.4.14 SetIpcPtpDsClockAccuracy

Description	Set the defaultDs.clockQuality.clockAccuracy parameter. In Master mode, the value of defaultDs.clockQuality.clockAccuracy is copied to the parentDs.grandmasterClockQuality.clockAccuracy and placed in the Announce packet grandmasterClockQuality.clockAccuracy field.
	`STATUS SetIpcPtpDsClockAccuracy(`
	`  IN int32 fnId,`
	`  IN u_char ptpClockAccuracy);`

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=SetIpcPtpDsClockAccuracy(0,41)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	ptpClockAccuracy	0 - 255	254

In profile 2 (G.8275.1) the user shall do the following:
SetIpcPtpDsClockAccuracy(0,41), 0x21, for a T-GM connected to a PRTC in locked-mode
SetIpcPtpDsClockAccuracy(0,254), 0xFE, for a T-GM not connected to a PRTC

5.3.4.15 GetIpcPtpDsClockAccuracy

Description	Get the defaultDs.clockQuality.clockAccuracy parameter.

	`STATUS GetIpcPtpDsClockAccuracy(`
	`  IN int32 fnId,`
	`  OUT u_char* ptpClockAccuracy);`

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=GetIpcPtpDsClockAccuracy(0,&ptpClockAccuracy)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	ptpClockAccuracy	See SetIpcPtpDsClockAccuracy	254

5.3.4.16 SetIpcPtpDsOffsetScaledLogVar

Description	Set the defaultDs.clockQuality.offsetScaledLogVar parameter. The value of defaultDs.clockQuality.offsetScaledLogVar is placed in the Announce packet.

	`STATUS SetIpcPtpDsOffsetScaledLogVar(`
	`  IN int32 fnId,`
	`  IN Uint16 ptpOffsetScaledLogVar);`

Applicability	IPC9xxx, IPC17xx
Example	rv=SetIpcPtpDsOffsetScaledLogVar(0,20061)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	ptpOffsetScaledLogVar	0 - 65535	65535

In profile 2 (G.8275.1) the user shall do the following:
SetIpcPtpDsOffsetScaledLogVar(0,20061), 0x4E5D, for a T-GM connected to a PRTC in locked-mode
SetIpcPtpDsOffsetScaledLogVar(0,65535), 0xFFFF, for a T-GM not connected to a PRTC

5.3.4.17 GetIpcPtpDsOffsetScaledLogVar

Description	Get the defaultDs.clockQuality.offsetScaledLogVar parameter.

	`STATUS GetIpcPtpDsOffsetScaledLogVar(`
	`  IN int32 fnId,`
	`  OUT Uint16* ptpOffsetScaledLogVar);`

Applicability	IPC9xxx, IPC17xx
Example	rv=GetIpcPtpDsOffsetScaledLogVar(0,&ptpOffsetScaledLogVar)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	ptpOffsetScaledLogVar	See SetIpcPtpDsOffsetScaledLogVar	65535

5.3.4.18 SetIpcPtpDsPriority1

Description	Set the defaultDs.priority1 parameter. The value of defaultDs.priority1 is placed in the Announce packet grandmasterPriority1 field.

	`STATUS SetIpcPtpDsPriority1(`
	`  IN int32 fnId,`
	`  IN int32 ptpPriority1);`

Applicability	IPC9xxx, IPC17xx
Example	rv=SetIpcPtpDsPriority1(0,100)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	ptpPriority1	0 - 255	255 Slave 128 Master 128 BC

5.3.4.19 GetIpcPtpDsPriority1

Description	Get the defaultDs.priority1 parameter.
	`STATUS GetIpcPtpDsPriority1(`
	`  IN int32 fnId,`
	`  OUT int32* ptpPriority1);`

Applicability	IPC9xxx, IPC17xx
Example	rv=GetIpcPtpDsPriority1(0,&ptpPriority1)

Parameters

Parameter Name	Range	Default
fnId	0	0
ptpPriority1	See SetIpcPtpDsPriority1	

5.3.4.20 SetIpcPtpDsPriority2

Description	Set the defaultDs.priority2 parameter. The value of defaultDS.priority2 is placed in the Announce packet grandmasterPriority2 field.
	`STATUS SetIpcPtpDsPriority2(`
	`  IN int32 fnId,`
	`  IN int32 ptpPriority2);`

Applicability	IPC9xxx, IPC17xx
Example	rv=SetIpcPtpDsPriority2(0,100)

Parameters

Parameter Name	Range	Default
fnId	0	0
ptpPriority2	0 - 255	255 Slave 128 Master 128 BC

5.3.4.21 GetIpcPtpDsPriority2

Description	Get the defaultDs.priority2 parameter.
	`STATUS GetIpcPtpDsPriority2(`
	`  IN int32 fnId,`
	`  OUT int32* ptpPriority2);`

Applicability	IPC9xxx, IPC17xx
Example	rv=GetIpcPtpDsPriority2(0,&ptpPriority2)

Parameters

Parameter Name	Range	Default
fnId	0	0
ptpPriority2	See SetIpcPtpDsPriority2	

5.3.4.22 SetIpcPtpDsDomainNumber

Description	Set the defaultDs.domainNumber parameter. The value of defaultDs.domainNumber is placed in all IEEE 1588 v2 packets header domainNumber field.
	`STATUS SetIpcPtpDsDomainNumber(`
	`  IN int32 fnId,`
	`  IN u_char ptpDomainNumber);`

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=SetIpcPtpDsDomainNumber(0,2)

Parameters

Parameter Name	Range	Default
fnId	0	0
ptpDomainNumber	N/A	0 (profile-0) 4 (profile-1) 24 (profile-2)

Note *SetIpcPtpAdminProfile (used of setting profile-0 or profile-1)set the ptpDomainNumber automatically. In case user set the ptpDomainNumber to none default value, it shall be done after using SetIpcPtpAdminProfile.*

5.3.4.23 GetIpcPtpDsDomainNumber

Description	Get the defaultDs.domainNumber parameter.
	`STATUS GetIpcPtpDsDomainNumber(`
	`  IN int32 fnId,`
	`  OUT u_char* ptpDomainNumber);`

Applicability	IPC9xxx, IPC17xx, IPC1603		
Example	rv=GetIpcPtpDsDomainNumber(0,&ptpDomainNumber)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	ptpDomainNumber	See SetIpcPtpDsDomainNumber	

5.3.4.24 GetIpcPtpDsSO

Description	Get the defaultDs.slaveOnly parameter.		
	STATUS GetIpcPtpDsSO(**IN int32 fnId,** **OUT int32* slaveOnly);**		
Applicability	IPC9xxx, IPC17xx, IPC1603		
Example	rv=GetIpcPtpDsSO(0,&slaveOnly)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	slaveOnly	0,1	1 Slave 0 Master 0 BC

5.3.4.25 SetIpcPtpDsLocalPriority

Description	Set the defaultDs.localPriority parameter. (G.8275.1)		
	STATUS SetIpcPtpDsLocalPriority(**IN int32 fnId,** **IN u_char localPriority);**		
Applicability	IPC9xxx, IPC17xx		
Example	rv=SetIpcPtpDsLocalPriority(0,100)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	localPriority	0 - 255	128 Slave 128 Master 128 BC

Note *localPriority applicable for ptpBmcMode=10*

5.3.4.26 GetIpcPtpDsLocalPriority

Description	Get the defaultDs.localPriority parameter. (G.8275.1)		
	STATUS GetIpcPtpDsLocalPriority(**IN int32 fnId,** **OUT u_char* localPriority);**		
Applicability	IPC9xxx, IPC17xx		
Example	rv=GetIpcPtpDsLocalPriority(0,&localPriority)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	localPriority	See SetIpcPtpDsLocalPriority	

5.3.5 Current Dataset

5.3.5.1 GetIpcPtpDsCurrent

Description	Get CurrentDs data.		
	STATUS GetIpcPtpDsCurrent(**IN int32 fnId,** **OUT CurrentDs_t* currentDs);**		
Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603		
Example	rv=GetIpcPtpDsCurrent(0,&currentDs)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	currentDs	N/A	N/A

currentDS shall be updated only if device slaveStateEx as reported by GetIpcPtpState is 8.

5.3.5.1.1 Default Values

The default parameters in currentDS are provided below.

Parameter	Master	Slave	BC
Steps removed	0	0	0
Offset From Master	0	0	0
Mean Path Delay	0	0	0

5.3.5.2 SetIpcPtpDsCurrentMode

Description	Set the currentDs mode of operation while in BC mode. Auto GM-CurDs: The currentDs.stepsRemoved set by the stepsRemoved +1 provided by the selected master (GM – grandmaster) as been defined by the BMC. Manual: The currentDs.stepsRemoved set manually using the SetIpcPtpDsStepsRemoved API.

```
STATUS SetIpcPtpDsCurrentMode(
    IN int32 fnId,
    IN int32 dsCurrentMode);
```

Applicability	IPC9xxx BC mode
Example	rv=SetIpcPtpDsCurrentMode(0,1)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	dsCurrentMode	0 – Auto GM-CurDs 1 – Manual	0

5.3.5.3 GetIpcPtpDsCurrentMode

Description	Get the currentDs mode of operation while in BC mode.

```
STATUS GetIpcPtpDsCurrentMode(
    IN int32 fnId,
    OUT int32* dsCurrentMode);
```

Applicability	IPC9xxx BC mode
Example	rv=GetIpcPtpDsCurrentMode(0,&dsCurrentMode)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	dsCurrentMode	See SetIpcPtpDsCurrentMode	

5.3.5.4 SetIpcPtpDsStepsRemoved

Description	Set the CurrentDs.stepsRemoved.

```
STATUS SetIpcPtpDsStepsRemoved(
    IN int32 fnId,
    IN int16 stepsRemoved);
```

Applicability	IPC9xxx, IPC17xx
Example	rv=SetIpcPtpDsStepsRemoved(0,2)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	stepsRemoved	0 - 255	0

While in BC modes, in order to use GetIpcPtpDsStepsRemoved, dsCurrentMode shall be set to 1.

5.3.5.5 GetIpcPtpDsStepsRemoved

Description	Get the CurrentDs.stepsRemoved parameter.

```
STATUS GetIpcPtpDsStepsRemoved(
    IN int32 fnId,
    OUT int16* stepsRemoved);
```

Applicability	IPC9xxx, IPC17xx		
Example	rv=GetIpcPtpDsStepsRemoved(0,&stepsRemoved)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	stepsRemoved	See SetIpcPtpDsStepsRemoved	

5.3.6 Parent Dataset

5.3.6.1 GetIpcPtpDsParent

Description	Get parentDs data.
	`STATUS GetIpcPtpDsParent(`
	`  IN int32 fnId,`
	`  OUT ParentDs_t* parentDs);`		

Applicability	IPC9xxx, IPC17xx, IPC1603		
Example	rv=GetIpcPtpDsParent(0,0,&parentDs)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	parentDs	N/A	N/A

5.3.6.1.1 Default Values

The default parameters in parentDs are provided in the below table:

Parameter	Master	Slave	BC
Port Identity	00:0A:35:FF:FE:01:F4:15:0000	00:0A:35:FF:FE:01:F4:14:0000	00:0A:35:FF:FE:01:F4:16:0000
Stats	0	0	0
Observed Parent:			
OffsetScaledLogVar	65535 (0xFFFF)	65535 (0xFFFF)	65535 (0xFFFF)
ClockPhaseChangeRate	2147483647 (0x7FFFFFFF)	2147483647 (0x7FFFFFFF)	2147483647 (0x7FFFFFFF)
Grandmaster:			
Identity	00:0A:35:FF:FE:01:F4:15	00:0A:35:FF:FE:01:F4:14	00:0A:35:FF:FE:01:F4:16
Clock Quality:			
ClockClass	248	255	248
ClockAccuracy	254	254	254
OffsetScaledLogVar	65535 (profile-0) 65535 (profile-1) 32768 (profile-2)	65535 (profile-0) 65535 (profile-1) 32768 (profile-2)	65535 (profile-0) 65535 (profile-1) 65535 (profile-2)
Priority 1	128	255	128
Priority 2	128	255	128
Local Priority	128	128	128

Note *Observed Parent:OffsetScaledLogVar – obsParOffsetScaledLogVar*
Observed Parent:ClockPhaseChangeRate – obsParClockPhaseChangeRate

Note *Local Priority applicable for ptpBmcMode=10*

5.3.6.2 SetIpcPtpAdminDsParentGm

Description	In IPC17xx – Master mode BC (masterMode=1) set by SetIpcPtpAdminMasterMode:
	Set the parent dataset grandmaster's parameters.
	`STATUS SetIpcPtpAdminDsParentGm(`
	`  IN int32  fnId,`
	`  IN char*  gmIdentity,`
	`  IN u_char gmCqClockClass,`
	`  IN u_char gmCqClockAccuracy,`
	`  IN Uint16 gmCqoffsetScaledLogVariance,`
	`  IN Uint32 gmPriority1,`
	`  IN Uint32 gmPriority2);`

Applicability	IPC17xx – Master mode BC
Example	IPC9xxx: Setting parent ClockClass rv=SetIpcPtpAdminDsParentGm(0,"0",50,0,0,0,0) IPC17xx: Setting GM parameters: rv=SetIpcPtpAdminDsParentGm(0,"00:0A:11:FF:FE:22:33:44",6,20,65535,100,111)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	gmIdentity	N/A	N/A
	gmCqClockClass	According to standard and profiles	See parentDs default
	gmCqClockAccuracy	According to standard	See parentDs default
	gmCqoffsetScaledLogVariance	N/A	See parentDs default
	gmPriority1	0 – 254	See parentDs default
	gmPriority2	0 – 254	See parentDs default

In IPC1703, SetIpcPtpAdminDsParentGm can be used only masterMode=1 (BC) as been set by SetIpcPtpAdminMasterMode
In IPC9xxx, SetIpcPtpAdminDsParentGm can be used only in BC mode

Parameter Name	***ParentDs Counterpart***
gmIdentity	*ParentDs.grandmasterIdentity*
gmCqClockClass	*ParentDs.grandmasterClockQuality.clockClass*
gmCqClockAccuracy	*ParentDs.grandmasterClockQuality.clockAccuracy*
gmCqOffsetScaledLogVariance	*ParentDs.grandmasterClockQuality.offsetScaledLogVar*
gmPriority1	*ParentDs.grandmasterPriority1*
gmPriority2	*ParentDs.grandmasterPriority2*

5.3.6.3 SetIpcPtpDsParentMode

Description: Set the parentDs mode of operation while in BC mode.
This API is applicable for the following contents:
parentDs.clockQuality.clockClass
parentDs.grandmasterPriority1
parentDs.grandmasterPriority2
parentDs.grandmasterClockQuality.clockAccuracy
parentDs.grandmasterClockQuality.offsetScaledLogVar

Manual-ParDs	The parentDs.clockQuality.clockClass is been set manually using SetIpcPtpDsParentPar API. The other 4 parentDs fields are been taken from defaultDs.
Auto	The parentDs is been set automatically.
Auto-A	The parentDs is been set automatically. Applicable for dsDefaultMode=4. HoSpec logic performed on ParentDs.

```
STATUS SetIpcPtpDsParentMode(
    IN int32 fnId,
    IN int32 dsParentMode);
```

Applicability: IPC9xxx BC mode

Example: rv=SetIpcPtpDsParentMode(0,6)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	dsParentMode	4 – Manual-ParDs 6 – Auto 7 – Auto-A	6

parentDs.clockQuality.clockClass for IPC9xxx-BC mode, dsParentMode=6,7:

Description	*Gstate.state*	*HO Spec*	*ClockClass*	
			startMode=4	*startMode=24*
FR	*1*	*N/A*	*248*	*248*
HO	*2*	*Out of spec*	*248*	*165*
HO	*2*	*In spec*	*248*	*135*
TR	*3*	*N/A*	*Previous state ClockClass*	*Previous state ClockClass*
LK	*4*	*N/A*	*GM*	*GM*

5.3.6.4 GetIpcPtpDsParentMode

Description: Get the parentDs mode of operation while in BC mode.

```
STATUS GetIpcPtpDsParentMode(
```

```
    IN int32 fnId,
    OUT int32* dsParentMode);
```

Applicability	IPC9xxx BC mode
Example	rv=GetIpcPtpDsParentMode(0,&dsParentMode)

Parameters

Parameter Name	Range	Default
fnId	0	0
dsParentMode	See SetIpcPtpDsParentMode	

5.3.6.5 SetIpcPtpDsParentPar

Description: Set the parentDs.clockQuality.clockClass parameter in case of dsParentMode=4. In case dsParentMode<>4, this API doesn't modified the parentDs.

```
STATUS SetIpcPtpDsParentPar(
    IN int32 fnId,
    IN u_char parentDsClockClass);
```

Applicability	IPC9xxx BC mode
Example	rv=SetIpcPtpDsParentPar(0,1)

Parameters

Parameter Name	Range	Default
fnId	0	0
parentDsClockClass	See SetIpcPtpDsClockClass	248

Note *This API is not stored in ConfDB.*

5.3.6.6 GetIpcPtpDsParentPar

Description: Get the parentDsClockClass as been set by SetIpcPtpDsParent.

```
STATUS GetIpcPtpDsParentPar(
    IN int32 fnId,
    OUT int32* parentDsClockClass);
```

Applicability	IPC9xxx BC mode
Example	rv=GetIpcPtpDsParentPar(0,&parentDsClockClass)

Parameters

Parameter Name	Range	Default
fnId	0	0
parentDsClockClass	See SetIpcPtpDsClockClass	248

5.3.6.7 SetIpcPtpDsParentClockPhaseChangeRate

Description: Set the parentDs.obsParClockPhaseChangeRate parameter.

```
STATUS SetIpcPtpDsParentClockPhaseChangeRate(
    IN int32 fnId,
    IN int32 ptpClockPhaseChangeRate);
```

Applicability	IPC9xxx, IPC17xx
Example	rv=SetIpcPtpDsParentClockPhaseChangeRate(0,100)

Parameters

Parameter Name	Range	Default
fnId	0	0
ptpClockPhaseChangeRate	-2147483646 – +2147483647	2147483647

Note *This API is not stored in ConfDB.*

5.3.6.8 GetIpcPtpDsParentClockPhaseChangeRate

Description: Get the parentDs.obsParClockPhaseChangeRate parameter.

```
STATUS GetIpcPtpDsParentClockPhaseChangeRate(
    IN int32 fnId,
    OUT int32* ptpClockPhaseChangeRate);
```

Applicability	IPC9xxx, IPC17xx
Example	rv=GetIpcPtpDsParentClockPhaseChangeRate(0,&ptpClockPhaseChangeRate)

Parameters

Parameter Name	Range	Default
fnId	0	0
ptpClockPhaseChangeRate	See SetIpcPtpDsParentClockPhaseChangeRate	2147483647

5.3.6.9 SetIpcPtpDsParentOffsetScaledLogVar

Description	Set the parentDs.obsParOffsetScaledLogVar parameter.
	`STATUS SetIpcPtpDsParentOffsetScaledLogVar(`
	`  IN int32 fnId,`
	`  IN int32 obsParOffsetScaledLogVar);`

Applicability	IPC9xxx, IPC17xx
Example	rv=SetIpcPtpDsParentOffsetScaledLogVar(0,100)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	obsParOffsetScaledLogVar	-65534 – +65535	65535

Note *This API is not stored in ConfDB.*

5.3.6.10 GetIpcPtpDsParentOffsetScaledLogVar

Description	Get the parentDs.obsParOffsetScaledLogVar parameter.
	`STATUS GetIpcPtpDsParentOffsetScaledLogVar(`
	`  IN int32 fnId,`
	`  OUT int32* obsParOffsetScaledLogVar);`

Applicability	IPC9xxx, IPC17xx
Example	rv=GetIpcPtpDsParentOffsetScaledLogVar(0,&obsParOffsetScaledLogVar)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	obsParOffsetScaledLogVar	See SetIpcPtpDsParentOffsetScaledLogVar	65535

5.3.6.11 SetIpcPtpDsParentStats

Description	Set the parentDs.ParentStats parameter.
	`STATUS SetIpcPtpDsParentStats(`
	`  IN int32 fnId,`
	`  IN int32 ptpParentStats);`

Applicability	IPC9xxx, IPC17xx
Example	rv=SetIpcPtpDsParentStats(0,1)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	ptpParentStats	0 – Not valid 1 – Valid	0

Note *The ptpParentStats is not part of the PTP header.*

Note *This API is not stored in ConfDB.*

5.3.6.12 GetIpcPtpDsParentStats

Description	Get the parentDs.ParentStats parameter.
	`STATUS GetIpcPtpDsParentStats(`
	`  IN int32 fnId,`
	`  OUT int32* ptpParentStats);`

Applicability	IPC9xxx, IPC17xx
Example	rv=GetIpcPtpDsParentStats(0,&ptpParentStats)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	ptpParentStats	See SetIpcPtpDsParentStats	0

5.3.7 Time Properties Dataset

5.3.7.1 GetIpcPtpDsTimeProperties

Description	Get timePropertiesDS data.
	`STATUS GetIpcPtpDsTimeProperties(`
	`  IN int32 fnId,`
	`  OUT TimePropertiesDs_t* timePropertiesDs);`

Applicability	IPC9xxx, IPC17xx
Example	rv=GetIpcPtpDsTimeProperties(0,&timePropertiesDs)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	timePropertiesDs	N/A	N/A

5.3.7.1.1 Default Values

The default parameters in timePropertiesDS are provided below.

Parameter	Master	Slave	BC
UTC offset	35	35	35
UTC offset valid	0	0	0
Leap 59	0	0	0
Leap 61	0	0	0
Time Traceable	0	0	0
Frequency Traceable	0	0	0
PTP Timescale	1	1	1
Time Source	160	160	160

5.3.7.2 SetIpcPtpDsTimePropertiesMode

Description	Set the timePropertiesDS mode of operation while in BC mode.
	Manual – The timePropertiesDs can be set manually using the timePropertiesDs APIs.
	Manual-GM – The timePropertiesDs.frequencyTraceable, only, can be set manually using the SetIpcPtpDsFrequencyTraceable API. Other properties are been controlled automatically based on GM.
	Auto – The timePropertiesDs is been controlled automatically.
	`STATUS SetIpcPtpDsTimePropertiesMode(`
	`  IN int32 fnId,`
	`  IN int32 dsTimePropertiesMode);`

Applicability	IPC9xxx BC mode
Example	rv=SetIpcPtpDsTimePropertiesMode(0,6)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	dsTimePropertiesMode	4 – Manual 5 – Manual-GM 6 – Auto	6

The timePropertiesDS in master mode, is been controlled directly by the following APIs (e.g. SetIpcPtpDsCurrentUtcOfs, SetIpcPtpDsCurrentUtcOfsValid etc.').

5.3.7.3 GetIpcPtpDsTimePropertiesMode

Description	Get the defaultDs mode of operation while in BC mode.
	`STATUS GetIpcPtpDsTimePropertiesMode(`
	`  IN int32 fnId,`
	`  OUT int32* dsTimePropertiesMode);`

Applicability	IPC9xxx BC mode
Example	rv=GetIpcPtpDsTimePropertiesMode(0,&dsTimePropertiesMode)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	dsTimePropertiesMode	See SetIpcPtpDsTimePropertiesMode	

5.3.7.4 SetIpcPtpDsCurrentUtcOfs

Description	Set the timePropertiesDS.currentUtcOffset parameter in order to determine the UTC.
	In startMode=0,2,7,20,22 (master): The value of timePropertiesDS.currentUtcOffset as being set by the API and is placed in the Announce packet currentUtcOffset field.
	In startMode=1,3,8,10,23 (slave): The value of timePropertiesDS.currentUtcOffset updates using the incoming information from the parent Master as reported by the Announce packets.
	In case the device is in HO, the timePropertiesDs maintains the parameters as were reported by the master before hangout.
	In case of non-available parent master or in HO, the API sets the timePropertiesDS.currentUtcOffset.
	In startMode=4,24 (BC): The value of timePropertiesDS.currentUtcOffset depends on dsTimePropertiesMode.
	-- dsTimePropertiesMode6: The value of timePropertiesDS.currentUtcOffset updates using the incoming information from the parent Master as reported by the Announce packets.
	In case the device is in HO, the timePropertiesDs shall maintained the parameters as were reported by the master before hangout.
	In case of non-available parent master or in HO, the API sets the timePropertiesDS.currentUtcOffset.
	-- dsTimePropertiesMode=4,5: The value of timePropertiesDS.currentUtcOffset as being set by the API used by the device.
	`STATUS SetIpcPtpDsCurrentUtcOfs(`
	`  IN int32 fnId,`
	`  IN int32 ptpCurrentUtcOffset);`

Applicability	IPC9xxx, IPC17xx
Example	rv=SetIpcPtpDsCurrentUtcOfs(0,36)

Parameters

Parameter Name	Range	Default
fnId	0	0
ptpCurrentUtcOffset	N/A	36

5.3.7.5 GetIpcPtpDsCurrentUtcOfs

Description	Get the timePropertiesDS.currentUtcOffset parameter.
	`STATUS GetIpcPtpDsCurrentUtcOfs(`
	`  IN int32 fnId,`
	`  OUT int32* ptpCurrentUtcOffset);`

Applicability	IPC9xxx, IPC17xx
Example	rv=GetIpcPtpDsCurrentUtcOfs(0,&ptpCurrentUtcOffset)

Parameters

Parameter Name	Range	Default
fnId	0	0
ptpCurrentUtcOffset	N/A	36

5.3.7.6 SetIpcPtpDsCurrentUtcOfsMode

Description	Set the currentUtcOffsetMode.
	ptpCurrentUtcOffsetMode=0: The value of timePropertiesDS.currentUtcOffset updates using the incoming information from the parent Master as reported by the Announce packets.
	ptpCurrentUtcOffsetMode=1: The timePropertiesDs.currentUtcOffset can be set manually using the SetIpcPtpDsCurrentUtcOfs.
	`STATUS SetIpcPtpDsCurrentUtcOfsMode(`
	`  IN int32 fnId,`
	`  IN int32 ptpCurrentUtcOffsetMode);`

Applicability	IPC9xxx Slave mode, IPC17xx Slave mode
Example	rv=SetIpcPtpDsCurrentUtcOfsMode(0,0)

Parameters

Parameter Name	Range	Default
fnId	0	0

	ptpCurrentUtcOffsetMode	0 – Auto 1 – Manual	0 – IPC9xxx 0 – IPC1703 1 – IPC17xx

5.3.7.7 GetIpcPtpDsCurrentUtcOfsMode

Description	Get the currentUtcOffseMode.
	`STATUS GetIpcPtpDsCurrentUtcOfsMode(`
	`IN int32 fnId,`
	`OUT int32* ptpCurrentUtcOffsetMode);`

Applicability	IPC9xxx Slave mode, IPC17xx Slave mode
Example	rv=GetIpcPtpDsCurrentUtcOfsMode(0,&ptpCurrentUtcOffsetMode)

Parameters

Parameter Name	Range	Default
fnId	0	0
ptpCurrentUtcOffsetMode	See SetIpcPtpDsCurrentUtcOfsMode	

5.3.7.8 SetIpcPtpDsCurrentUtcOfsValid

Description	Set the timePropertiesDS.currentUtcOffsetValid parameter. The value of timePropertiesDS.currentUtcOffsetValid is placed in the Announce packet header's flagField currentUtcOffsetValid bit.
	`STATUS SetIpcPtpDsCurrentUtcOfsValid(`
	`IN int32 fnId,`
	`IN int32 ptpCurrentUtcOffsetValidFlag);`

Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode
Example	rv=SetIpcPtpDsCurrentUtcOfsValid(0,1)

Parameters

Parameter Name	Range		Default
fnId	0		N/A
ptpCurrentUtcOffsetValidFlag	0	FALSE	0
	1	TRUE	

5.3.7.9 GetIpcPtpDsCurrentUtcOfsValid

Description	Get the timePropertiesDS.currentUtcOffsetValid parameter.
	`STATUS GetIpcPtpDsCurrentUtcOfsValid(`
	`IN int32 fnId,`
	`OUT int32* ptpCurrentUtcOffsetValidFlag);`

Applicability	IPC9xxx, IPC17xx
Example	rv=GetIpcPtpDsCurrentUtcOfsValid(0,&ptpCurrentUtcOffsetValidFlag)

Parameters

Parameter Name	Range		Default
fnId	0		0
ptpCurrentUtcOffsetValidFlag	See	SetIpcPtpDsCurrentUtcOfsValid	0

5.3.7.10 SetIpcPtpDsLeap59

Description	Set timePropertiesDS.leap59 parameter. The value of timePropertiesDS.leap59 is placed in the Announce packet header flagField leap59 bit.
	`STATUS SetIpcPtpDsLeap59(`
	`IN int32 fnId,`
	`IN int32 ptpLeap59Flag);`

Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode
Example	rv=SetIpcPtpDsLeap59(0,1)

Parameters

Parameter Name	Range		Default
fnId	0		0
ptpLeap59Flag	0	FALSE	0
	1	TRUE	

5.3.7.11 GetIpcPtpDsLeap59

Description	Get timePropertiesDS.leap59 parameter.
	`STATUS GetIpcPtpDsLeap59(`
	`IN int32 fnId,`
	`OUT int32* ptpLeap59Flag);`

Applicability	IPC9xxx, IPC17xx			
Example	rv=GetIpcPtpDsLeap59(0,&ptpLeap59Flag)			
Parameters	Parameter Name		Range	Default
	fnId	0		0
	ptpLeap59Flag	See	SetIpcPtpDsLeap59	0

5.3.7.12 SetIpcPtpDsLeap61

Description	Set timePropertiesDS.leap61 parameter. The value of timePropertiesDS.leap61 is placed in the Announce packet header flagField leap61 bit.			
	STATUS SetIpcPtpDsLeap61(**IN int32 fnId,** **IN int32 ptpLeap61Flag);**			
Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode			
Example	rv=SetIpcPtpDsLeap61(0,1)			
Parameters	Parameter Name		Range	Default
	fnId	0		0
	ptpLeap61Flag	0	FALSE	0
		1	TRUE	

5.3.7.13 GetIpcPtpDsLeap61

Description	Get timePropertiesDS.leap61 parameter.			
	STATUS GetIpcPtpDsLeap61(**IN int32 fnId,** **OUT int32* ptpLeap61Flag);**			
Applicability	IPC9xxx, IPC17xx			
Example	rv=GetIpcPtpDsLeap61(0,&ptpLeap61Flag)			
Parameters	Parameter Name		Range	Default
	fnId	0		0
	ptpLeap61Flag	See	SetIpcPtpDsLeap61	0

5.3.7.14 SetIpcPtpDsTimeTraceable

Description	Set the timePropertiesDS.timeTraceable parameter. The value of timePropertiesDS.timeTraceable is placed in the Announce packet header's flagField timeTraceable bit.			
	STATUS SetIpcPtpDsTimeTraceable(**IN int32 fnId,** **IN int32 ptpTimeTraceableFlag);**			
Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode			
Example	rv=SetIpcPtpDsTimeTraceable(0,1)			
Parameters	Parameter Name		Range	Default
	fnId	0		0
	ptpTimeTraceableFlag	0	FALSE	0
		1	TRUE	

5.3.7.15 GetIpcPtpDsTimeTraceable

Description	Get the timePropertiesDS.timeTraceable parameter.			
	STATUS GetIpcPtpDsTimeTraceable(**IN int32 fnId,** **OUT int32* ptpTimeTraceableFlag);**			
Applicability	IPC9xxx, IPC17xx			
Example	rv=GetIpcPtpDsTimeTraceable(0,&ptpTimeTraceableFlag)			
Parameters	Parameter Name		Range	Default
	fnId	0		0
	ptpTimeTraceableFlag	See	SetIpcPtpDsTimeTraceable	0

5.3.7.16 SetIpcPtpDsFrequencyTraceable

Description	This API set the timePropertiesDS.frequencyTraceable.

	Applicable for master modes:
	clkInRefMode=4 – set by SetIpcAdminClkInRefMode
	Applicable for BC modes:
	dsTimePropertiesMode=4,5 – set by SetIpcPtpDsTimePropertiesMode
	`STATUS SetIpcPtpDsFrequencyTraceable(`
	`IN int32 fnId,`
	`IN int32 ptpFrequencyTraceableFlag);`

Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode
Example	rv=SetIpcPtpDsFrequencyTraceable(0,1)

Parameters	Parameter Name			Range	Default
	fnId	0			0
	ptpFrequencyTraceableFlag	0	FALSE		0
		1	TRUE		

5.3.7.17 GetIpcPtpDsFrequencyTraceable

Description	Set the timePropertiesDS.frequencyTraceable parameter.
	`STATUS GetIpcPtpDsFrequencyTraceable(`
	`IN int32 fnId,`
	`OUT int32* ptpFrequencyTraceableFlag);`

Applicability	IPC9xxx, IPC17xx
Example	rv=GetIpcPtpDsFrequencyTraceable(0,&ptpFrequencyTraceableFlag)

Parameters	Parameter Name		Range	Default
	fnId	0		0
	ptpFrequencyTraceableFlag	See	SetIpcPtpDsFrequencyTraceable	0

5.3.7.18 SetIpcPtpDsTimescale

Description	Set the timePropertiesDS.ptpTimescale parameter. The value of timePropertiesDS.timeTraceable is placed in the Announce packet header's flagField ptpTimescale bit.
	`STATUS SetIpcPtpDsTimescale(`
	`IN int32 fnId,`
	`IN int32 ptpTimescaleFlag);`

Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode
Example	rv=SetIpcPtpDsTimescale(0,1)

Parameters	Parameter Name			Range	Default
	fnId	0			0
	ptpTimescaleFlag	0	ARB		1
		1	PTP		

5.3.7.19 GetIpcPtpDsTimescale

Description	Get the timePropertiesDS.ptpTimescale parameter. The value of timePropertiesDS.timeTraceable is placed in the Announce packet header's flagField ptpTimescale bit.
	`STATUS GetIpcPtpDsTimescale(`
	`IN int32 fnId,`
	`OUT int32* ptpTimescaleFlag);`

Applicability	IPC9xxx, IPC17xx
Example	rv=GetIpcPtpDsTimescale(0,&ptpTimescaleFlag)

Parameters	Parameter Name		Range	Default
	fnId	0		0
	ptpTimescaleFlag	See	SetIpcPtpDsTimescale	1

5.3.7.20 SetIpcPtpDsTimeSource

Description	Set the timePropertiesDS.timeSource parameter. The value of timePropertiesDS.timeSource is placed in the Announce packet timeSource field.
	`STATUS SetIpcPtpDsTimeSource(`
	`IN int32 fnId,`
	`IN u_char ptpTimeSource);`

Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode			
Example	rv=SetIpcPtpDsTimeSource(0,32)			
Parameters	Parameter Name		Range	Default
	fnId	0		0
	ptpTimeSource	16	Atomic clock	160
		32	GPS	
		48	Terrestrial Radio	
		64	PTP	
		80	NTP	
		96	Hand set	
		144	Other	
		160	Internal oscillator	

5.3.7.21 GetIpcPtpDsTimeSource

Description	Get the timePropertiesDS.timeSource parameter.			
	STATUS GetIpcPtpDsTimeSource(**IN int32 fnId,** **OUT u_char* ptpTimeSource);**			
Applicability	IPC9xxx, IPC17xx			
Example	rv=GetIpcPtpDsTimeSource(0,&ptpTimeSource)			
Parameters	Parameter Name		Range	Default
	fnId	0		0
	ptpTimeSource	See SetIpcPtpDsTimeSource		160

5.3.7.22 GetIpcPtpDsUtcProp

Description	Get utcPropDs data. The utcPropDs is part of the timePropertiesDS.			
	STATUS GetIpcPtpDsUtcProp(**IN int32 fnId,** **OUT UtcProperties* utcPropDs);**			
Applicability	IPC9xxx, IPC17xx			
Example	rv=GetIpcPtpDsUtcProp(0,&utcPropDs)			
Parameters	Parameter Name		Range	Default
	fnId	0		0
	utcPropDs	N/A		N/A

5.3.7.23 GetIpcPtpDsTraceProp

Description	Get tracePropDs data. The tracePropDs is part of the timePropertiesDS.			
	STATUS GetIpcPtpDsTraceProp(**IN int32 fnId,** **OUT TraceabilityProperties* tracePropDs);**			
Applicability	IPC9xxx, IPC17xx			
Example	rv=GetIpcPtpDsTraceProp(0,&tracePropDs)			
Parameters	Parameter Name		Range	Default
	fnId	0		0
	tracePropDs	N/A		N/A

5.3.7.24 GetIpcPtpDsTimeProp

Description	Get timePropDs data. The timePropDs is part of the timePropertiesDS.			
	STATUS GetIpcPtpDsTimeProp(**IN int32 fnId,** **OUT TimescaleProperties* timePropDs);**			
Applicability	IPC9xxx, IPC17xx			
Example	rv=GetIpcPtpDsTimeProp(0,&timePropDs)			
Parameters	Parameter Name		Range	Default
	fnId	0		0
	timePropDs	N/A		N/A

5.3.8 Port Dataset

5.3.8.1 GetIpcPtpDsPort

Description	Get portDs data.
	`STATUS GetIpcPtpDsPort(`
	`  IN int32 fnId,`
	`  IN Uint16 portIndex,`
	`  OUT PortDs* portDs);`		

Applicability	IPC9xxx, IPC17xx, IPC1603		
Example	rv=GetIpcPtpDsPort(0,1,&portDs)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	portIndex	portMapMode=0 1 BC-Slave port 2 BC-Master port 1 Slave mode 2 Master mode portMapMode=>1 1-N Specific port	N/A
	portDs	N/A	N/A

Note *N = ptpNumberPorts*

5.3.8.1.1 Default Values

The default parameters in portDs are provided below.

Parameter	Master	Slave	BC
Port Identity	00:0A:35:FF:FE:01:F4:15 :0002	00:0A:35:FF:FE:01:F4:14 :0001	S: 00:0A:35:FF:FE:01:F4:16:0001 M:00:0A:35:FF:FE:01:F4:16:0002 M:00:0A:35:FF:FE:01:F4:16:0003 M:00:0A:35:FF:FE:01:F4:16:0004 M:00:0A:35:FF:FE:01:F4:16:0005 ... M:00:0A:35:FF:FE:01:F4:16:000A .. M:00:0A:35:FF:FE:01:F4:16:0041 (0x41 = 65)
Port State	6	9	S:9 M:6
Log Min Delay Req Interval	-7 (profile-0,1) -4 (profile-2)	-3 (profile-0,1) -4 (profile-2)	S:-3 (profile-0,1) M:-5 (profile-0,1) S:-4 (profile-2) M:-4 (profile-2)
Peer Mean Path Delay	0	0	0
Log Announce Interval	1 (profile-0,1) -3 (profile-2)	1 (profile-0,1) -3 (profile-2)	1 (profile-0,1) -3 (profile-2)
Announce Receipt Timeout	5 (profile-0,1) 3 (profile-2)	5 (profile-0,1) 3 (profile-2)	5 (profile-0,1) 3 (profile-2)
Log Sync Interval	-5 (profile-0,1) -4 (profile-2)	-5 (profile-0,1) -4 (profile-2)	-5 (profile-0,1) -4 (profile-2)
Delay Mechanism	1	1	1
Log Min PDelay Req Interval	-5	-5	-5
Version Number	2	2	2
localPriority	128	128	128
notSlave	1	0	PortMapMode=0 S(PI=1): 0 M(PI=2..64): 1 PortMapMode=>1 PI=1..64: 0

Note *Calculation of logMinDelayReqInterval for Slave port (portIndex=1): logMinDelayReqInterval=-LOG2(GetIpcPtpAdminRxSyncRate/GetIpcPtpAdminDelayReqInterval).*

5.3.8.2 GetIpcPtpDsPortReport

Description	Get the report of PortDs. Retrieve according to portIndex order.
	`STATUS GetIpcPtpDsPortReport(`
	`IN int32 fnId,`
	`IN int32 page,`
	`OUT PortDS_s* portDsReport);`

Applicability	IPC9xxx, IPC17xx
Example	rv=GetIpcPtpDsPortReport(0,1,&portDsReport)

Parameters

Parameter Name	Range	Default
fnId	0	0
page	0 CI 1-16 1 CI 17-32 2 CI 33-48 3 CI 49-64	0
DsPortReport	N/A	N/A

Note *STATUS=-7 in case of empty page.*

5.3.8.3 GetIpcPtpDsPortReportAll

Description	Get the report of PortDs. Report size is PORT_INDEX_MAX. Retrieve according to portIndex order.
	`STATUS GetIpcPtpDsPortReportAll(`
	`IN int32 fnId,`
	`OUT PortDSAll_s* portDsReportAll);`

Applicability	Host mode : IPC9xxx, IPC17xx
Example	rv=GetIpcPtpDsPortReportAll(0,&portDsReportAll)

Parameters

Parameter Name	Range	Default
fnId	0	0
portDsReportAll	N/A	N/A

5.3.8.4 GetIpcPtpDsPortState

Description	Get the portDs.portState.
	`STATUS GetIpcPtpDsPortState(`
	`IN int32 fnId,`
	`IN Uint16 portIndex,`
	`OUT u_char* portState);`

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=GetIpcPtpDsPortState(0,1,&ptpPortState)

Parameters

Parameter Name	Range	Default
fnId	0	0
portIndex	portMapMode=0 1 BC-Slave port 2 BC-Master port 1 Slave mode 2 Master mode portMapMode=>1 1-N Specific port	N/A
portState	1 INITIALIZING 2 FAULTY 3 DISBALED 4 LISTENING 5 PRE-MASTER 6 MASTER 7 PASSIVE 8 UNCALIBRATED 9 SLAVE	N/A

Note *N = ptpNumberPorts*

5.3.8.5 GetIpcPtpDsPortNum

Description	Get the PortDs.portIdentity.portNum. **STATUS GetIpcPtpDsPortNum(** **IN int32 fnId,** **IN Uint16 portIndex,** **OUT Uint16* portNum);**
Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=GetIpcPtpDsPortNum(0,1,&portNum)

Parameters

Parameter Name	Range	Default
fnId	0	0
portIndex	portMapMode=0 1 BC Slave port 2 BC-Master port 1 Slave mode 2 Master mode portMapMode=>1 1-N Specific ports	0
portNum	1…65535	Slave/Master 1 BC-Slave port 1 BC-Master port 2

Note *N = ptpNumberPorts*

5.3.8.6 SetIpcPtpDsPortNum

Description	Set the PortDs.portIdentity.portNum **STATUS SetIpcPtpDsPortNum(** **IN int32 fnId,** **IN Uint16 portIndex,** **IN Uint16 portNum);**
Applicability	IPC9xxx, IPC17xx
Example	rv=SetIpcPtpDsPortNum(0,1,10)

Parameters

Parameter Name	Range	Default
fnId	0	0
portIndex	portMapMode=0 1 BC-Slave port 2 BC-Master port 1 Slave mode 2 Master mode portMapMode=>1 1-N Specific port	N/A
portNum	1…65535	Slave/Master 1 BC Slave 1 BC Master 2

5.3.8.7 SetIpcPtpAnnounceReqIntervalLog

Description	Set the requested logInterMessagePeriod for Announce packet. The logInterMessagePeriod is been sent by the slave port RUTS (REQUEST UNICAST TRANSMISSION TLV) for Announce packets. **STATUS SetIpcPtpAnnounceReqIntervalLog(** **IN int32 fnId,** **IN int32 logIntrMsgPeriod);**
Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=SetIpcPtpAnnounceReqIntervalLog(0,2)

Parameters

Parameter Name	Range	Default
fnId	0	0
logIntrMsgPeriod	-3 .. 7	1

5.3.8.8 GetIpcPtpAnnounceReqIntervalLog

Description	Get the requested logInterMessagePeriod for Announce packet. The logInterMessagePeriod is been sent by the slave port RUTS (REQUEST UNICAST TRANSMISSION TLV) for Announce packets.

```
STATUS SetIpcPtpAnnounceReqIntervalLog(
    IN int32 fnId,
    OUT int32* logIntrMsgPeriod);
```

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=GetIpcPtpAnnounceReqIntervalLog(0,&logIntrMsgPeriod)

Parameters

Parameter Name	Range	Default
fnId	0	0
logIntrMsgPeriod	See SetIpcPtpAnnounceReqIntervalLog	

5.3.8.9 SetIpcPtpDsAnnounceRxTimeout

Description	Set the Announce Receipt Timeout in number of Announce intervals updating portDs.logAnnounceReceiptTimeout.

```
STATUS SetIpcPtpDsAnnounceRxTimeout(
    IN int32 fnId,
    IN u_char ptpAnnounceRxTimeout);
```

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=SetIpcPtpDsAnnounceRxTimeout(0,7)

Parameters

Parameter Name	Range	Default
fnId	0	0
ptpAnnounceRxTimeout	4-16 3	5 (profile-0,1) 3 (profile-2)

The expirationCounter (TTL) as reported by GetIpcPtpBmcKnownMasterTable (ForeignMaster.expirationCounter) is defined by ptpAnnounceIntervalLog and ptpAnnounceRxTimeout as follow:
expirationCounter = 2^ptpAnnounceIntervalLog x ptpAnnounceRxTimeout
IF expirationCounter <2 THEN expirationCounter=2

5.3.8.10 GetIpcPtpDsAnnounceRxTimeout

Description	Get the Announce Receipt Timeout in number of Announce intervals that updates portDs.logAnnounceReceiptTimeout.

```
STATUS GetIpcPtpDsAnnounceRxTimeout(
    IN int32 fnId,
    OUT u_char* ptpAnnounceRxTimeout);
```

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=GetIpcPtpDsAnnounceRxTimeout(0,&ptpAnnounceRxTimeout)

Parameters

Parameter Name	Range	Default
fnId	0	0
ptpAnnounceRxTimeout	See SetIpcPtpDsAnnounceRxTimeout	

5.3.8.11 SetIpcPtpMcastSyncRate

Description	Sets the Sync packet rate updating portDs.logSyncInterval parameter. The value of portDs.logSyncInterval is placed in the Sync packet header logMessageInterval field if sent in Multicast.

```
STATUS SetIpcPtpMcastSyncRate(
    IN int32 fnId,
    IN Uint32 logSyncInterval);
```

Applicability	IPC9xxx – Master mode, IPC17xx – Master mode
Example	rv=SetIpcPtpMcastSyncRate(0,-5)

Parameters

Parameter Name	Range	Default
fnId	0	0
logSyncInterval	-1…-7	-5 (startMode=2,7) -4 (startMode=22,24)

5.3.8.12 GetIpcPtpMcastSyncRate

Description	Get the portDs.logSyncInterval parameter.
	`STATUS GetIpcPtpMcastSyncRate(`
	`  IN int32 fnId,`
	`  OUT Uint32* logSyncInterval);`

Applicability	IPC9xxx – Master mode, IPC17xx – Master mode
Example	rv=GetIpcPtpMcastSyncRate(0,&logSyncInterval)

Parameters

Parameter Name	Range	Default
fnId	0	0
logSyncInterval	See SetIpcPtpMcastSyncRate	

5.3.8.13 GetIpcPtpDsVerNum

Description	Get the portDs.versionNumber parameter.
	`STATUS GetIpcPtpDsVerNum(`
	`  IN int32 fnId,`
	`  IN Uint16 portIndex,`
	`  OUT u_char* versionNumber);`

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=GetIpcPtpDsVerNum(0,1,&versionNumber)

Parameters

Parameter Name	Range	Default
fnId	0	0
portIndex	portMapMode=0 1 BC-Slave port 2 BC-Master port 1 OC-Slave mode 2 OC-Master mode portMapMode=>1 1-N Specific port	N/A
versionNumber	2	2

5.3.8.14 SetIpcPtpDsPortLocalPriority

Description	Set the portDs.localPriority parameter. (G.8275.1)
	`STATUS SetIpcPtpDsPortLocalPriority(`
	`  IN int32 fnId,`
	`  IN Uint16 portIndex`
	`  IN u_char localPriority);`

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=SetIpcPtpDsPortLocalPriority(0,1,2)

Parameters

Parameter Name	Range	Default
fnId	0	0
portIndex	portMapMode=0 1 BC-Slave port 2 BC-Master port 1 OC-Slave mode 2 OC-Master mode portMapMode=>1 0 All ports 1-N Specific port	N/A
localPriority	1 - 255	128

Note *localPriority applicable for ptpBmcMode=10*

5.3.8.15 GetIpcPtpDsPortLocalPriority

Description	Get the portDs.localPriority parameter. (G.8275.1)
	`STATUS GetIpcPtpDsPortLocalPriority(`
	`  IN int32 fnId,`
	`  IN Uint16 portIndex,`
	`  OUT u_char* localPriority);`

Applicability	IPC9xxx, IPC17xx, IPC1603		
Example	rv=GetIpcPtpDsPortLocalPriority(0,1,&localPriority)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	portIndex	portMapMode=0 1 BC-Slave port 2 BC-Master port 1 OC-Slave mode 2 OC-Master mode portMapMode=>1 1-N Specific port	N/A
	localPriority	1 - 255	128

5.3.8.16 SetIpcPtpDsPortNotSlave

Description	Set the portDs.NotSlave parameter. (G.8275.1) **STATUS SetIpcPtpDsPortNotSlave(** **IN int32 fnId,** **IN Uint16 portIndex** **IN u_char notSlave);**		
Applicability	IPC9xxx, IPC17xx, IPC1603		
Example	rv=SetIpcPtpDsPortNotSlave(0,1,0)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	portIndex	portMapMode=0 1 BC-Slave port 2 BC-Master port 1 OC-Slave mode 2 OC-Master mode portMapMode=>1 0 All ports 1-N Specific port	N/A
	notSlave	0 - 1	portMapMode=0 PI(1) – 0 PI(2) – 1 PI(3) – 1 …. PI(64) – 1 portMapMode=>1 PI(1) – 0 PI(2) – 0 PI(3) – 0 …. PI(64) – 0

Note *notSlave applicable for ptpBmcMode=10.*

5.3.8.17 GetIpcPtpDsPortNotSlave

Description	Get the portDs.NotSlave parameter. (G.8275.1) **STATUS GetIpcPtpDsPortNotSlave(** **IN int32 fnId,** **IN Uint16 portIndex,** **OUT u_char* notSlave);**		
Applicability	IPC9xxx, IPC17xx, IPC1603		
Example	rv=GetIpcPtpDsPortNotSlave(0,1,&notSlave)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	See	SetIpcPtpDsPortNotSlave	
	See	SetIpcPtpDsPortNotSlave	

5.3.9 Unicast

5.3.9.1 SetIpcPtpUnicastMasterAdd

Description	Add a unicast Master to a Slave unicastMasterTable

```
STATUS SetIpcPtpUnicastMasterAdd(
    IN int32 fnId,
    IN char* portAddress);
```

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=SetIpcPtpUnicastMasterAdd(0,“192.168.100.10”)

Parameters

Parameter Name	Range	Default
fnId	0	0
portAddress	“0.0.0.0” – “255.255.255.255”	N/A

Invoke this command to add a Master IP address to the Unicast Master Table. The Slave send REQUEST_UNICAT_TRANSMISSION_SIGNALLING messages to the Master (or masters) that are in the Unicast Master Table. The Unicast Master Table can contain up to eight Master IP addresses.

5.3.9.2 SetIpcPtpUnicastMasterDelete

Description	Delete a unicast Master from a Slave unicastMasterTable

```
STATUS SetIpcPtpUnicastMasterDelete(
    IN int32 fnId,
    IN char* portAddress);
```

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=SetIpcPtpUnicastMasterDelete(0,“192.168.100.10”)

Parameters

Parameter Name	Range	Default
fnId	0	0
portAddress	“0.0.0.0” – “255.255.255.255”	N/A

The Slave stop sending REQUEST_UNICAT_TRANSMISSION_SIGNALLING messages to the deleted Master. The Slave send CANCEL_UNICAST_TRANSMISSION messages for Announce, Sync and Delay Response.

5.3.9.3 SetIpcPtpUnicastMasterTableClr

Description	Clear unicastMasterTable.

```
STATUS SetIpcPtpUnicastMasterTableClr(
    IN int32 fnId);
```

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=SetIpcPtpUnicastMasterTableClr(0)

Parameters

Parameter Name	Range	Default
fnId	0	0

5.3.9.4 GetIpcPtpUnicastMasterTable

Description	Get the Slave’s unicastMasterTable. This API returns IP address.

```
STATUS GetIpcPtpUnicastMasterTable(
    IN int32 fnId,
    OUT SIeee1588PrfPortAddressQueryTable_t*
        ucastMasterTable);
```

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=GetIpcPtpUnicastMasterTable(0,&ucastMasterTable)

Parameters

Parameter Name	Range	Default
fnId	0	0
ucastMasterTable	N/A	Slave: 192.168.1.21 192.168.1.22 BC: 192.168.1.21

The ucastMasterTable default includes 192.168.1.21 and 192.168.1.22 in order to enable the device in Slave and BC modes to request unicast service from default Master and BC IP address.

5.3.9.5 SetIpcPtpUnicastDelFromUniTableEn

Description	Enable or disable delete unicast Master from a Slave unicastMasterTable upon

	receiving CANCEL_UNICAST_TRANSMISSION (An/Sync/DResp) **STATUS SetIpcPtpUnicastDelFromUniTableEn(** **IN int32 fnId,** **IN int32 delFromUniTableEn);**
Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=SetIpcPtpUnicastDelFromUniTableEn(0,1)

Parameters	Parameter Name	Range		Default
	fnId	0		0
	delFromUniTableEn	0	Disable	0
		1	Enable	

5.3.9.6 GetIpcPtpUnicastDelFromUniTableEn

Description	Get the Enable or disable of delete unicast Master from a Slave unicastMasterTable upon receiving CANCEL_UNICAST_TRANSMISSION (An/Sync/DResp) **STATUS GetIpcPtpUnicastDelFromUniTableEn(** **IN int32 fnId,** **OUT int32* delFromUniTableEn);**
Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=GetIpcPtpUnicastDelFromUniTableEn(0,&delFromUniTableEn)

Parameters	Parameter Name	Range		Default
	fnId	0		0
	delFromUniTableEn	See SetIpcPtpUnicastDelFromUniTableEn		0

5.3.9.7 GetIpcPtpUnicastAcceptableMasterTable

Description	Get the Slave's acceptableMasterTable. **STATUS GetIpcPtpUnicastAcceptableMasterTable(** **IN int32 fnId,** **OUT SIeee1588PrfAcceptableMasterTable_t*** **acceptMasterTable);**
Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=GetIpcPtpUnicastAcceptableMasterTable(0,&acceptMasterTable)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	acceptMasterTable	N/A	192.168.1.20, priority1 128 192.168.1.21, priority1 128

Note *The acceptMasterTable default includes 192.168.1.20 and 192.168.1.21 in order to enable IPC9xxx BC/Slave to request unicast service from default BC/Master IP address.*

5.3.9.8 GetIpcPtpUnicastAcceptableMasterTableExtended

Description	Get the Slave's acceptableMasterTableExtended. **STATUS GetIpcPtpUnicastAcceptableMasterTableExtended(** **IN int32 fnId,** **OUT SIeee1588PrfAcceptableMasterTableExtended_t*** **acceptMasterTableEx);**
Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=GetIpcPtpUnicastAcceptableMasterTableExtended(0,&acceptMasterTableEx)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	acceptMasterTableEx	N/A	192.168.1.20, priority1 128 192.168.1.21, priority1 128

Note *acceptMasterTableEx is superset of the acceptMasterTable.*

Note *This API returns large amount of data and consume significant resources of the management UART (mUART).*

5.3.9.9 SetIpcPtpUnicastAcceptableMasterTableEn

Description	Enable or disable the acceptableMasterTable.

	STATUS SetIpcPtpUnicastAcceptableMasterTableEn(**IN int32 fnId,** **IN int32 acceptableMasterTableEn);**
Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=SetIpcPtpUnicastAcceptableMasterTableEn(0,1)

Parameters

Parameter Name		Range	Default
fnId	0		0
acceptableMasterTableEn	0	Disable	0/1
	1	Enable	

Note *acceptableMasterTableEn is automatically controlled by SetIpcPtpAdminProfile.*

5.3.9.10 GetIpcPtpUnicastAcceptableMasterTableEn

Description	Get the acceptableMasterTable mode of operation.
	STATUS GetIpcPtpUnicastAcceptableMasterTableEn(**IN int32 fnId,** **OUT int32* acceptableMasterTableEn);**
Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=GetIpcPtpUnicastAcceptableMasterTableEn(0,&acceptableMasterTableEn)

Parameters

Parameter Name		Range	Default
fnId	0		0
acceptableMasterTableEn	See	SetIpcPtp…..TableEn	0

5.3.9.11 SetIpcPtpUnicastAcceptableMasterAdd

Description	Add a unicast Master to acceptableMasterTable.
	STATUS SetIpcPtpUnicastAcceptableMasterAdd(**IN int32 fnId,** **IN char* portAddress,** **IN int16 alternatePriority1);**
Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=SetIpcPtpUnicastAcceptableMasterAdd(0,“192.168.100.10”,8)

Parameters

Parameter Name	Range	Default
fnId	0	0
portAddress	“0.0.0.0” – “255.255.255.255”	N/A
alternatePriority1	0 – 255	0

5.3.9.12 SetIpcPtpUnicastAcceptableMasterDelete

Description	Delete a unicast Master acceptableMasterTable.
	STATUS SetIpcPtpUnicastAcceptableMasterDelete(**IN int32 fnId,** **IN char* portAddress);**
Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=SetIpcPtpUnicastAcceptableMasterDelete(0,“192.168.100.10”)

Parameters

Parameter Name	Range	Default
fnId	0	0
portAddress	“0.0.0.0” – “255.255.255.255”	N/A

5.3.9.13 SetIpcPtpUnicastAcceptableMasterTableClr

Description	Clear acceptableMasterTable.
	STATUS SetIpcPtpUnicastAcceptableMasterTableClr(**IN int32 fnId);**
Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=SetIpcPtpUnicastAcceptableMasterTableClr(0)

Parameters

Parameter Name	Range	Default
fnId	0	0

5.3.9.14 SetIpcPtpUnicastPtsfConfig

Description	Set the Slave's PTSF Configuration.
	`STATUS SetIpcPtpUnicastPtsfConfig(`
	`IN int32 fnId,`
	`IN int32 ptsfSyncTh,`
	`IN int32 ptsfAnnounceTh,`
	`IN int32 ptsfUnusableTh,`
	`IN int32 ptsfUnusablePdvTh`
	`IN int32 ptsfWaitToRestore);`

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=SetIpcPtpUnicastPtsfConfig(0,60,60,240,500,0)

Parameters

Parameter Name	Range	Default
fnId	0	0
ptsfSyncTh	1 – 1200 sec	60
ptsfAnnounceTh	1 – 1200 sec	60
ptsfUnusableTh	1 – 1200 sec	240
ptsfUnusablePdvTh	1 – 5000 us	500
ptsfWaitToRestore	0 – 100000 sec (0 – do not restore)	0

5.3.9.15 GetIpcPtpUnicastPtsfConfig

Description	Get the Slave's PTSF Configuration.
	`STATUS GetIpcPtpUnicastPtsfConfig(`
	`IN int32 fnId,`
	`OUT SIeee1588PrfUnicastPtsfConfig_t* ptsfConfig);`

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=GetIpcPtpUnicastPtsfConfig(0,&unicastPtsfConfig)

Parameters

Parameter Name	Range	Default
fnId	0	0
unicastPtsfConfig	See SetIpcPtpUnicastPtsfConfig	N/A

5.3.9.16 SetIpcPtpUnicastAcceptableMasterPtsfMode

Description	Set the unicast Acceptable Master PTSF fMode
	`STATUS SetIpcPtpUnicastAcceptableMasterPtsfMode(`
	`IN int32 fnId,`
	`IN char* portAddress,`
	`IN int32 ptsfMode);`

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=SetIpcPtpUnicastAcceptableMasterPtsfMode(0,"192.168.100.10",0)

Parameters

Parameter Name	Range	Default
fnId	0	0
portAddress	"0.0.0.0" – "255.255.255.255"	N/A
ptsfMode	0 – OFF (Pass or ignore) 1 – ON (Fail) 2 - AUTO	2

Note *GetIpcPtpUnicastAcceptableMasterTableExtended presents the ptsfMode.*

5.3.9.17 SetIpcPtpUnicastRutsEn

Description	Enable or disable RUTS (REQUEST UNICAST TRANSMISSION TLV) per packet type.
	`STATUS SetIpcPtpUnicastRutsEn(`
	`IN int32 fnId,`
	`IN int32 pktType,`
	`IN int32 rutsEn);`

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=SetIpcPtpUnicastRutsEn(0,2,0)

Parameters

Parameter Name	Range	Default
fnId	0	0

pktType	1	Sync	N/A
	2	Delay Response	
	4	Announce	
	7	All packets	
rutsEn	0	Disable	1
	1	Enable	

5.3.9.18 GetIpcPtpUnicastRutsEn

Description	Get RUTS (REQUEST UNICAST TRANSMISSION TLV) operation mode per packet type.
	`STATUS GetIpcPtpUnicastRutsEn(`
	`  IN int32 fnId,`
	`  OUT RutsEn* rutsEn);`

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=GetIpcPtpUnicastRutsEn(0,&rutsEn) GetIpcPtpUnicastRutsEn: 1 1 1

Parameters	Parameter Name	Range	Default
	fnId	0	0
	rutsEn	See SetIpcPtpUnicastRutsEn	1

5.3.9.19 SetIpcPtpUnicastRutsPeriod

Description	Set the RUTS (REQUEST UNICAST TRANSMISSION TLV) Tx period for all packet types.
	`STATUS SetIpcPtpUnicastRutsPeriod(`
	`  IN int32 fnId,`
	`  IN int16 rutsPeriod);`

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=SetIpcPtpUnicastRutsPeriod(0,32)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	rutsPeriod	1, 2, 4, 8, 16, 2^n, 2^{n+1}, 1024	8

The rutsPeriod define the maximal period between two successive RUTS (REQUEST UNICAST TRANSMISSION TLV) packets for the same packet type. The slave port shall send part of the RUTS packets with shorten period in order to improve unicast negotiation in case of large number of devices.

5.3.9.20 GetIpcPtpUnicastRutsPeriod

Description	Get the RUTS (REQUEST UNICAST TRANSMISSION TLV) Tx period for all packet types.
	`STATUS GetIpcPtpUnicastRutsPeriod(`
	`  IN int32 fnId,`
	`  OUT int16* rutsPeriod);`

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=GetIpcPtpUnicastRutsPeriod(0,&rutsPeriod)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	rutsPeriod	SetIpcPtpUnicastRutsPeriod	

5.3.9.21 SetIpcPtpUnicastRutsDuration

Description	Set the RUTS (REQUEST UNICAST TRANSMISSION TLV) durationField per packet type.
	`STATUS SetIpcPtpUnicastRutsDuration(`
	`  IN int32 fnId,`
	`  IN int32 rutsDurationAn,`
	`  IN int32 rutsDurationSy,`
	`  IN int32 rutsDurationDR);`

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=SetIpcPtpUnicastRutsDuration(0,300,300,300)

Parameters	Parameter Name	Range	Default

fnId	0	0
rutsDurationAn	10 – 1000	300
rutsDurationSy	10 – 1000	300
rutsDurationDR	10 – 1000	300

5.3.9.22 GetIpcPtpUnicastRutsDuration

Description	Get the RUTS (REQUEST UNICAST TRANSMISSION TLV) durationField per packet type.

```
STATUS GetIpcPtpUnicastRutsDuration(
    IN int32 fnId,
    OUT RutsDuration* rutsDuration);
```

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603		
Example	rv=GetIpcPtpUnicastRutsDuration(0,&rutsDuration)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	rutsDuration	See SetIpcPtpUnicastRutsDuration	

5.3.9.23 GetIpcPtpUnicastMasterTableMax

Description	Get the unicast master table max size.

```
STATUS GetIpcPtpUnicastMasterTableMax(
    IN int32 fnId,
    OUT int32* UniMasterTabMax);
```

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603		
Example	rv=GetIpcPtpUnicastMasterTableMax(0,&UniMasterTabMax)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	UniMasterTabMax	8	8

5.3.9.24 GetIpcPtpUnicastAcceptableMasterTableMax

Description	Get the unicast acceptable master table max size.

```
STATUS GetIpcPtpUnicastAcceptableMasterTableMax(
    IN int32 fnId,
    OUT int32* UniAccMasterTabMax);
```

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603		
Example	rv=GetIpcPtpUnicastAcceptableMasterTableMax(0,&UniAccMasterTabMax)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	UniAccMasterTabMax	8	8

5.3.9.25 GetIpcPtpAcceptableTableAll

Description	Get the acceptableTableAll.

```
STATUS GetIpcPtpAcceptableTableAll(
    IN int32 fnId,
    OUT AcceptableTableAll_t* acceptableTableAll);
```

Applicability	IPC9xxx, IPC17xx		
Example	rv=GetIpcPtpAcceptableTableAll(0,&acceptableTableAll)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	acceptableTableAll	N/A	See comment

Note *In Terminal mode – print only valid entries.*

Note *STATUS=-7 in case of empty table.*

5.3.9.26 GetIpcPtpAcceptableTable

Description	Get the Slave's acceptableTable.

```
STATUS GetIpcPtpAcceptableTable(
    IN int32 fnId,
    IN int32 page,
    OUT AcceptableTable_t* acceptableTable);
```

Applicability	IPC9xxx, IPC17xx
Example	rv=GetIpcPtpAcceptableTable(0,&acceptableTable)

Parameters

Parameter Name	Range		Default
fnId	0		0
page	0	1-16 addresses	
	1	17-32 addresses	
	2	33-48 addresses	
	3	49-64 addresses	
acceptableTable	N/A		See comment

Note *In Terminal mode – print only valid entries.*

Note *STATUS=-7 in case of empty page.*

The acceptableTable enable varies defaults in order to enable IPC9xxx BC, Slave and Master to plug and play while using default IP/MAC address of BC, Slave and Master.

Note

portMapMode=1:
BC mode:
portIndex=1: 00:0A:35:01:F4:15 or 192.168.1.21
portIndex=2: 00:0A:35:01:F4:14 or 192.168.1.20
Master mode:
portIndex=1: 00:0A:35:01:F4:16 or 192.168.1.22
portIndex=2: 00:0A:35:01:F4:14 or 192.168.1.20
Slave mode:
portIndex=1: 00:0A:35:01:F4:16 or 192.168.1.22
portIndex=2: 00:0A:35:01:F4:15 or 192.168.1.21

portMapMode=2:
BC mode:
portIndex=1: 00:0A:35:01:F4:15 or 192.168.1.21
portIndex=2: 00:0A:35:01:F4:14 or 192.168.1.20
Master mode:
portIndex=1: 00:0A:35:01:F4:16 or 192.168.1.22
Slave mode:
portIndex=1: 00:0A:35:01:F4:16 or 192.168.1.22

5.3.9.27 SetIpcPtpAcceptableTableAdd

Description	Add address acceptableTable.

```
STATUS SetIpcPtpAcceptableTableAdd(
    IN int32 fnId,
    IN char* addr,
    IN Uint16 portIndex;
```

Applicability	IPC9xxx, IPC17xx
Example	startMode = 23,24 rv=SetIpcPtpAcceptableTableAdd(0,“00:0A:35:01:F4:15”,1) startMode = 1,3,4,8,10 rv=SetIpcPtpAcceptableTableAdd(0,“192.168.1.21”,1)

Parameters

Parameter Name	Range	Default
fnId	0	0
addr	“00:00:00:00:00:00” – “FF:FF:FF:FF:FF:FF” “0.0.0.0” – “255.255.255.255”	N/A
portIndex	1 – ptpNumberPorts	N/A

5.3.9.28 SetIpcPtpAcceptableTableDelete

Description	Delete address from acceptableTable

```
STATUS SetIpcPtpAcceptableTableDelete(
```

	IN int32 fnId, **IN char* addr);**
Applicability	IPC9xxx, IPC17xx
Example	startMode = 23,24 rv=SetIpcPtpAcceptableTableDelete(0,“00:0A:35:01:F4:15”) startMode = 1,3,4,8,10 rv=SetIpcPtpAcceptableTableDelete(0,“192.168.1.21”)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	addr	“00:00:00:00:00:00” – “FF:FF:FF:FF:FF:FF” “0.0.0.0” – “255.255.255.255”	N/A

5.3.9.29 SetIpcPtpAcceptableTableClr

Description	Clear acceptableTable. **STATUS SetIpcPtpAcceptableTableClr(** **IN int32 fnId);**
Applicability	IPC9xxx, IPC17xx
Example	rv=SetIpcPtpAcceptableTableClr(0)

Parameters	Parameter Name	Range	Default
	fnId	0	0

5.3.10 Packet Counters

5.3.10.1 SetIpcPtpCntPortMode

Description	Set the PTP packet counters mode. **STATUS SetIpcPtpCntPortMode(** **IN int32 fnId,** **IN Uint16 CntPortMode);**
Applicability	IPC9xxx
Example	rv=SetIpcPtpCntPortMode(0,0)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	CntPortMode	0 – Disabled 1 – Enabled	1

5.3.10.2 GetIpcPtpCntPortMode

Description	Get the PTP packet counters mode. **STATUS GetIpcPtpCntPortMode(** **IN int32 fnId,** **OUT int32* cntPortMode);**
Applicability	IPC9xxx
Example	rv=GetIpcPtpCntPortMode(0,&cntPortMode)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	cntPortMode	See SetIpcPtpCntPortMode	0

5.3.10.3 GetIpcPtpCntPort

Description	Get the PTP packet counters per port. **STATUS GetIpcPtpCntPort(** **IN int32 fnId,** **IN Uint16 portIndex,** **OUT PacketCntPort_t* packetCntPort);**
Applicability	IPC9xxx
Example	rv=GetIpcPtpCntPort(0,1,&packetCntPort)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	portIndex	portMapMode=0 1 BC-Slave port	N/A

		2 BC-Master port 1 OC-Slave mode 2 OC-Master mode portMapMode=1 1-N Specific port	
	packetCntPort	N/A	N/A

Packet counters count the packets received or transmitted by the device PTP protocol stack. In case the device trying to transmit packets, while ARP was not succeeded to provide the physical address, the applicable transmit counter will be increased, but the packet will not be transmitted by the device.

5.3.10.4 SetIpcPtpCntPortReportClr

Description	Clear the PTP packet counters report.
	`STATUS SetIpcPtpCntPortReportClr(`
	`IN int32 fnId,`
	`IN Uint16 portIndex);`		

Applicability	IPC9xxx		
Example	rv=SetIpcPtpCntPortReportClr(0,0)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	portIndex	portMapMode=0 1 BC-Slave port 2 BC-Master port 1 OC-Slave mode 2 OC-Master mode portMapMode!=0 0 All ports 1-N Specific port	N/A

N = ptpNumberPorts

5.3.10.5 GetIpcPtpCntPortReport

Description	Get the PTP packet counters port report.
	`STATUS GetIpcPtpCntPortReport(`
	`IN int32 fnId,`
	`IN int32 page,`
	`OUT PacketCntPort_s* packetCntPortRep);`		

Applicability	IPC9xxx		
Example	rv=GetIpcPtpCntPortReport(0,1,&packetCntPortRep)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	page	0 PI 1-16 1 PI 17-32 2 PI 33-48 3 PI 49-64	0
	packetCntPortRep	N/A	N/A

In terminal mode:
Modulo(1000000): TxSync, RxSync, TxFU, RxFU, TxDReq, RxDReq, TxDResp, RxDResp
Modulo(100000): others counters

Packet counters count the packets received or transmitted by the device PTP protocol stack. In case the device trying to transmit packets, while ARP was not succeeded to provide the physical address, the applicable transmit counter will be increased, but the packet will not be transmitted by the device.

5.3.10.6 GetIpcPtpCntPortReportAggr

Description	Get the PTP packet counters port aggregated report.
	`STATUS GetIpcPtpCntPortReportAggr(`
	`IN int32 fnId,`
	`OUT PacketCntPort_t* packetCntPortAggr);`

Applicability	IPC9xxx
Example	rv=GetIpcPtpCntPortReportAggr(0,&packetCntPortAggr)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	packetCntPortAggr	N/A	N/A

Packet counters count the packets received or transmitted by the device PTP protocol stack. In case the device trying to transmit packets, while ARP was not succeeded to provide the physical address, the applicable transmit counter will be increased, but the packet will not be transmitted by the device.

5.3.10.7 GetIpcPtpCntOther

Description	Get the PTP other packet counter (packetCntOther). The packetCntOther. RxNonMapped count the AcceptableTable non-mapped packets.

```
STATUS GetIpcPtpCntOther(
    IN int32 fnId,
    OUT PacketCntOther_t* packetCntOther);
```

Applicability	IPC9xxx
Example	rv=GetIpcPtpCntOther(0,&packetCntOther)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	packetCntOther	N/A	N/A

5.3.10.8 SetIpcPtpCntOtherClr

Description	Clear the PTP packet counters report (packetCntOther).

```
STATUS SetIpcPtpCntOtherClr(
    IN int32 fnId);
```

Applicability	IPC9xxx
Example	rv=SetIpcPtpCntOtherClr(0)

Parameters	Parameter Name	Range	Default
	fnId	0	0

5.3.10.9 GetIpcPtpCntRepSlave

Description	Get the PTP packet counters report.

```
STATUS GetIpcPtpCntRepSlave(
    IN int32 fnId,
    OUT PacketCntSlave* packetCntSlave);
```

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=GetIpcPtpCntRepSlave(0,&packetCntSlave)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	packetCntSlave	N/A	0

5.3.10.10 SetIpcPtpCntRepSlaveClr

Description	Clear the PTP packet counters report.

```
STATUS SetIpcPtpCntRepSlaveClr(
    IN int32 fnId);
```

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=SetIpcPtpCntRepSlaveClr(0)

Parameters	Parameter Name	Range	Default
	fnId	0	0

5.3.10.11 GetIpcPtpCntRepMaster

Description	Get the PTP packet counters report.

```
STATUS GetIpcPtpCntRepMaster(
    IN int32 fnId,
    OUT PacketCntMaster* packetCntMaster);
```

Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode
Example	rv=GetIpcPtpCntRepMaster(0,&packetCntMaster)

Parameters	Parameter Name	Range	Default
	fnId	0	0

	packetCntMaster	N/A	0

5.3.10.12 SetIpcPtpCntRepMasterClr

Description	Clear the PTP packet counters report.
	`STATUS SetIpcPtpCntRepMasterClr(`
	`  IN int32 fnId);`

Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode
Example	rv=SetIpcPtpCntRepMasterClr(0)

Parameters	Parameter Name	Range	Default
	fnId	0	0

5.3.10.13 GetIpcPtpCntRepSignaling

Description	Get the signaling packet counters report.
	`STATUS GetIpcPtpCntRepSinagling(`
	`  IN int32 fnId,`
	`  OUT PacketCntSignaling* packetCntSignaling);`

Applicability	IPC9xxx, IPC17xx
Example	rv=GetIpcPtpCntRepSignaling(0,&packetCntSignaling)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	packetCntSignaling	N/A	0

5.3.10.14 GetIpcPtpCntRepSignalingLastMin

Description	Get the signaling last minute packet counters report.
	`STATUS GetIpcPtpCntRepSinaglingLastMin(`
	`  IN int32 fnId,`
	`  OUT PacketCntSignalingLastMin*`
	`packetCntSignalingLastMin);`

Applicability	IPC9xxx, IPC17xx
Example	rv=GetIpcPtpCntRepSignalingLastMin(0,&packetCntSignalingLastMin)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	packetCntSignalingLastMin	N/A	0

5.3.10.15 SetIpcPtpCntRepSignalingClr

Description	Clear the signaling and signaling last minute packet counters report.
	`STATUS SetIpcPtpCntRepSignalingClr(`
	`  IN int32 fnId);`

Applicability	IPC9xxx, IPC17xx
Example	rv=SetIpcPtpCntRepSignalingClr(0)

Parameters	Parameter Name	Range	Default
	fnId	0	0

5.3.10.16 GetIpcPtpCntTxSyncCi

Description	Get the number of transmitted Sync packets for a slave with channel index CI. This API is applicable for unicast channels only.
	`STATUS GetIpcPtpCntTxSyncCi(`
	`  IN int32 fnId,`
	`  IN Uint32 masterCi,`
	`  OUT Uint32* masterTxSyncCnt);`

Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode
Example	rv=GetIpcPtpCntTxSyncCi(0,6,&masterTxSyncCnt)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	masterCi	0-63	N/A
	masterTxSyncCnt	N/A	0

5.3.10.17 SetIpcPtpCntTxSyncCiClr

Description	Reset the transmitted sync packet counter for a slave with channel index CI. This API is applicable for unicast channels only.

```
STATUS SetIpcPtpCntTxSyncCiClr(
    IN int32 fnId,
    IN Uint32 masterCi);
```

Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode
Example	rv=SetIpcPtpCntTxSyncCiClr(0,6)

Parameters

Parameter Name	Range	Default
fnId	0	0
masterCi	0-63	N/A

5.3.10.18 GetIpcPtpCntTxFollowUpCi

Description	Get the number of transmitted Follow up packets for a slave with channel index CI. This API is applicable for unicast channels only.

```
STATUS GetIpcPtpCntTxFollowUpCi(
    IN int32 fnId,
    IN Uint32 masterCi,
    OUT Uint32* masterTxFollowupCnt);
```

Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode
Example	rv=GetIpcPtpCntTxFollowUpCi(0,6,&masterTxFollowupCnt)

Parameters

Parameter Name	Range	Default
fnId	0	0
masterCi	0-63	N/A
masterTxFollowupCnt	N/A	0

5.3.10.19 SetIpcPtpCntTxFollowUpCiClr

Description	Reset the transmitted Follow up packet counter for a slave with channel index CI. This API is applicable for unicast channels only.

```
STATUS SetIpcPtpCntTxFollowUpCiClr(
    IN int32 fnId,
    IN Uint32 masterCi);
```

Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode
Example	rv=SetIpcPtpCntTxFollowUpCiClr(0,6)

Parameters

Parameter Name	Range	Default
fnId	0	0
masterCi	0-63	N/A

5.3.10.20 GetIpcPtpCntRxDelayReqCi

Description	Get the received Delay Request packet counter value for a slave with channel index CI. This API is applicable for unicast channels only.

```
STATUS GetIpcPtpCntRxDelayReqCi(
    IN int32 fnId,
    IN Uint32 masterCi,
    OUT Uint32* masterRxDelayReqCnt);
```

Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode
Example	rv=GetIpcPtpCntRxDelayReqCi(0,6,&masterRxDelayReqCnt)

Parameters

Parameter Name	Range	Default
fnId	0	0
masterCi	0-63	N/A
masterRxDelayReqCnt	N/A	0

5.3.10.21 SetIpcPtpCntRxDelayReqCiClr

Description	Reset the Delay Request packets counter for a slave with channel index CI. This API is applicable for unicast channels only.

```
STATUS SetIpcPtpCntRxDelayReqCiClr(
    IN int32 fnId,
    IN Uint32 masterCi);
```

Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode
Example	rv=SetIpcPtpCntRxDelayReqCiClr(0,6)

Parameters

Parameter Name	Range	Default
fnId	0	0
masterCi	0-63	N/A

5.3.10.22 GetIpcPtpCntRxDelayReqMissingCi

Description: Get the number of missing Delay Request packets counter value for a slave with channel index CI. Normally, a missing packet is a packet dropped by the network and did not reach the Slave. This API is applicable for unicast channels only.

```
STATUS GetIpcPtpCntRxDelayReqMissingCi(
    IN int32 fnId,
    IN Uint32 masterCi,
    OUT Uint32* masterDelReqMissCnt);
```

Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode
Example	rv=GetIpcPtpCntRxDelayReqMissingCi(0,6,&masterDelReqMissCnt)

Parameters

Parameter Name	Range	Default
fnId	0	0
masterCi	0-63	N/A
masterDelReqMissCnt	N/A	0

5.3.10.23 GetIpcPtpCntRxDelayReqMissorderCi

Description: Get the number of miss ordered Delay Request packets counter value for a slave with channel index CI. Miss order packets are packets with sequence number that are not monotonically increasing. This API is applicable for unicast channels only.

```
STATUS GetIpcPtpCntRxDelayReqMissorderCi(
    IN int32 fnId,
    IN Uint32 masterCi,
    OUT Uint32* masterDelReqMissOrderCnt);
```

Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode
Example	rv=GetIpcPtpCntRxDelayReqMissorderCi(0,6,&masterDelReqMissOrderCnt)

Parameters

Parameter Name	Range	Default
fnId	0	0
masterCi	0-63	N/A
masterDelReqMissOrderCnt	N/A	0

5.3.10.24 SetIpcPtpCntRxDelayReqMissCiClr

Description: Reset the PtpCntRxDelayReqMissingCi and the PtpCntRxDelayReqMissorderCi counters for a slave with channel index CI. This API is applicable for unicast channels only.

```
STATUS SetIpcPtpCntRxDelayReqMissCiClr(
    IN int32 fnId,
    IN Uint32 masterCi);
```

Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode
Example	rv=SetIpcPtpCntRxDelayReqMissCiClr(0,6)

Parameters

Parameter Name	Range	Default
fnId	0	0
masterCi	0-63	N/A

5.3.10.25 GetIpcPtpCntTxDelayRespCi

Description: Get the number of transmitted Delay Response packets for a slave with channel index CI. This API is applicable for unicast channels only.

```
STATUS GetIpcPtpCntTxDelayRespCi(
    IN int32 fnId,
    IN Uint32 masterCi,
    OUT Uint32* masterTxDelayRespCnt);
```

Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode
Example	rv=GetIpcPtpCntTxDelayRespCi(0,6,&masterTxDelayRespCnt)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	masterCi	0-63	N/A
	masterTxDelayRespCnt	N/A	0

5.3.10.26 SetIpcPtpCntTxDelayRespCiClr

Description	Reset the PtpCntRxDelayReqMissingCi and the PtpCntRxDelayReqMissorderCi counters for a slave with channel index CI. This API is applicable for unicast channels only.

```
STATUS SetIpcPtpCntTxDelayRespCiClr(
    IN int32 fnId,
    IN Uint32 masterCi);
```

Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode
Example	rv=SetIpcPtpCntTxDelayRespCiClr(0,6)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	masterCi	0-63	N/A

5.3.11 Best Master Clock (BMC)

5.3.11.1 SetIpcPtpBmcMode

Description	Set the Best Master Clock Algorithm (BMCA) mode.

```
STATUS SetIpcPtpBmcMode(
    IN int32 fnId,
    IN int32 ptpBmcMode);
```

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=SetIpcPtpBmcMode(0,1)

Parameters	Parameter Name		Range	Default
	fnId	0		0
	ptpBmcMode	0	Disable BMC	1 – profile 0
		1	Enable default BMC (IEEE 1588-2008, Annex J - chapter: J.3 Delay Request-Response Default PTP profile)	2 – profile 1 10 – profile 2
		2	Enable Alternate BMC (ITU-T G.8265.1)	
		10	Enable Alternate BMC (ITU-T G.8275.1)	

Note *ptpBmcMode is automatically controlled by SetIpcPtpAdminProfile.*

Note *PTP default profile (IEEE 1588-2008, Annex J - chapter: J.3 Delay Request-Response Default PTP profile).*

Note *EDStatus.ED_ST is provided by the GetIpcAdminExtDeviceStatus API per A/B.*

5.3.11.2 GetIpcPtpBmcMode

Description	Get the Best Master Clock Algorithm (BMCA) mode.

```
STATUS GetIpcPtpBmcMode(
    IN int32 fnId,
    OUT int32* ptpBmcMode);
```

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=GetIpcPtpBmcMode(0,&ptpBmcMode)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	ptpBmcMode	See SetIpcPtpBmcMode	

5.3.11.3 SetIpcPtpBmcModeEx

Description	Set the Best Master Clock Algorithm (BMCA) Extended mode.

```
STATUS SetIpcPtpBmcModeEx(
```

```
    IN  int32 fnId,
    IN  int32 ignoreUniMasterTable,
    IN  int32 msg,
    IN  int32 fourcePrio1_255,
    IN  int32 etiUpdateInterval,
    IN  int32 delSel,
    IN  int32 stepsRemovedMax,
    IN  int32 clsPortNumMode,
    IN  int32 ignoreParent_Ho_HoSpecDis,
    IN  int32 report);
```

Applicability: IPC9xxx – BC/Slave modes

Example: rv=SetIpcPtpBmcModeEx(0,0,1,0,10,3,255,1,0,2)

Parameters:

Parameter Name	Range	Default
fnId	0	0
ignoreUniMasterTable	0 – Use UnicastMasterTable	0
	1 – Ignore UnicastMasterTable	
msg	0 – Inform “Master Aged from bmcKnownMasterTable”	1
	1 – Do not inform “Master Aged from bmcKnownMasterTable”	
fourcePrio1_255	0 – Tx Priority1 after GM aged as per DefaultDS	0
	1 – Tx Priority1 after GM aged of 255	
etiUpdateInterval	ETI update interval in sec	10
delSel	Delay Selection	4
stepsRemovedMax	10 – 255	255
clsPortNumMode	0 – Disable setting of clsPortNum by BMC	1
	1 – Enable setting of clsPortNum by BMC in startMode=23,24	
	2 – Enable setting of clsPortNum by BMC in all startMode	
ignoreParent_Ho_ HoSpecDis	0	0
report	2	2

Note

Default of ignoreUniMasterTable:
IF startMode = 1,3,4 THEN ignoreUniMasterTable=0
IF startMode = 8,10,23,24 THEN ignoreUniMasterTable=1

Note

In order to change the device settings, the following sequence shall be performed after invoking the API:
SetIpcAdminStore
SetIpcPtpAdminStop
SetIpcPtpAdminStart
Or
SetIpcAdminStore
SetIpcPtpAdminReset

5.3.11.4 GetIpcPtpBmcModeEx

Description: Get the Best Master Clock Algorithm (BMCA) Extended mode.

```
STATUS GetIpcPtpBmcModeEx(
    IN  int32 fnId,
    OUT ptpBmcModeEx* ptpBmcModeEx);
```

Applicability: IPC9xxx – BC/Slave modes

Example: rv=GetIpcPtpBmcModeEx(0,&ptpBmcModeEx)

Parameters:

Parameter Name	Range	Default
fnId	0	0
ptpBmcModeEx	See SetIpcPtpBmcModeEx	N/A

5.3.11.5 SetIpcPtpBmcIgnoreUniMasterTable

Description: Set the ignoreUniMasterTable.

```
STATUS SetIpcPtpBmcIgnoreUniMasterTable(
    IN int32 fnId,
```

	IN int32 ignoreUniMasterTable);
Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=SetIpcPtpBmcIgnoreUniMasterTable(0,1)

Parameters

Parameter Name		Range	Default
fnId	0		0
ignoreUniMasterTable	0	Use UnicastMasterTable	0
	1	Ignore UnicastMasterTable	

Note *Control the ptpBmcModeEx.ignoreUniMasterTable.*

Note *Default of ignoreUniMasterTable:*
IF startMode = 1,3,4 THEN ignoreUniMasterTable=0
IF startMode = 8,10,23,24 THEN ignoreUniMasterTable=1

Note *In order to change the device settings, the following sequence shall be performed after invoking the API:*
SetIpcAdminStore
SetIpcPtpAdminStop
SetIpcPtpAdminStart
Or
SetIpcAdminStore
SetIpcPtpAdminReset

5.3.11.6 GetIpcPtpBmcIgnoreUniMasterTable

Description	Get the ignoreUniMasterTable.

```
STATUS GetIpcPtpBmcIgnoreUniMasterTable(
    IN int32 fnId,
    OUT int32* ignoreUniMasterTable);
```

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=GetIpcPtpBmcIgnoreUniMasterTable(0,&ignoreUniMasterTable)

Parameters

Parameter Name	Range	Default
fnId	0	0
ignoreUniMasterTable	See SetIpcPtpBmcIgnoreUniMasterTable	

5.3.11.7 SetIpcPtpBmcMasterManSel

Description	Manually sets the selected Master IP/MAC/portIdentity address.

```
STATUS SetIpcPtpBmcMasterManSel(
    IN int32 fnId,
    IN char* masterAddr);
```

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=SetIpcPtpBmcMasterManSel(0,“192.168.1.21”) rv=SetIpcPtpBmcMasterManSel(0,“00:0A:35:01:F4:16”) rv=SetIpcPtpBmcMasterManSel(0,”00:0A:35:FF:FE:01:F4:16:0002”)

Parameters

Parameter Name	Range	Default
fnId	0	0
masterAddr	N/A	N/A

Note *Using this API shall automatically disable BMC.*

Note *startMode = 1,3,4,8,10 – Master IP address setting shall be used.*
startMode = 23,24 – Master MAC address or portIdentity setting shall be used.

Note clockIdentity shall be in accordance with IEEE EUI-48 as per IEEE 1588 standard.

5.3.11.8 GetIpcPtpBmcMasterSelected

Description	Get the selected Master IP/MAC address.

```
STATUS GetIpcPtpBmcMasterSelected(
    IN int32 fnId,
    OUT char** masterAddr);
```

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603

Example	rv=GetIpcPtpBmcMasterSelected(0,&masterAddr)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	masterAddr	N/A	N/A

Note *In case master was not selected, the IP address 192.168.0.0 considered as default selected master. 192.168.0.0 used for local communications within a private network as specified by RFC 1918.*

Note *In case master was not selected, the MAC address FC:C2:40:00:00:00 considered as default selected master.*

Note *In case master was not selected, and aptsState=1, and the External Time Input (ETI) port was selected, the GetIpcPtpBmcMasterSelected return: [GetIpcPtpBmcMasterSelected]: 192.168.0.0 (ETI).*

5.3.11.9 GetIpcPtpBmcMasterSelectedEx

Description	Get the selected Master SelectedMasterEx. This API applicable only for ptpBmcMode !=0.		
	STATUS GetIpcPtpBmcMasterSelectedEx(**IN int32 fnId,** **OUT SelectedMasterEx_t* selectedMasterEx);**		
Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603		
Example	rv=GetIpcPtpBmcMasterSelectedEx(0,&selectedMasterEx)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	selectedMasterEx	N/A	N/A

Note *The parameters source:*

```
grandmasterIdentity     // GetIpcPtpDsParent
parentPortIdentity      // GetIpcPtpDsParent
parentUniAddr           // GetIpcPtpDsParent
portNum                 // local mapping
portIndex               // local mapping
```

5.3.11.10 GetIpcPtpBmcMasterSelectedUniReport

Description	Get the masters selected unicast report.		
	STATUS GetIpcPtpBmcMasterSelectedUniReport(**IN int32 fnId,** **OUT SelectedMasterUniRep_t* selectedMasterUniRep);**		
Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode		
Example	rv=GetIpcPtpBmcMasterSelectedUniReport(0,&selectedMasterUniRep)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	selectedMasterUniRep	N/A	N/A

5.3.11.11 GetIpcPtpBmcKnownMasterTable

Description	Get the table of known masters (foreginMasterDS).		
	STATUS GetIpcPtpBmcKnownMasterTable(**IN int32 fnId,** **OUT ForeignMasterMainTable_t* bmcKnownMasterTable);**		
Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603		
Example	rv=GetIpcPtpBmcKnownMasterTable(0,&bmcKnownMasterTable)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	bmcKnownMasterTable	N/A	N/A

5.3.12 State Decision Algorithm (SDA)

5.3.12.1 SetIpcPtpSdaMode

Description	Set the State Decision Algorithm (SDA) mode.
	STATUS SetIpcPtpSdaMode(

```
    IN int32 fnId,
    IN int32 sdaMode);
```

Applicability	IPC9xxx			
Example	rv=SetIpcPtpSdaMode(0,1)			
Parameters	Parameter Name		Range	Default
	fnId	0		0
	sdaMode	0	Disable SDA	0/1
		1	Enable SDA	

Default:
portMapMode=0 – sdaMode=0
portMapMode!=0 – sdaMode=1

5.3.12.2 GetIpcPtpSdaMode

Description	Get the State Decision Algorithm (SDA) mode.			

	`STATUS GetIpcPtpSdaMode(`
	`  IN int32 fnId,`
	`  OUT int32* sdaMode);`			

Applicability	IPC9xxx			
Example	rv=GetIpcPtpSdaMode(0,&sdaMode)			
Parameters	Parameter Name		Range	Default
	fnId	0		0
	sdaMode	See SetIpcPtpSdaMode		

5.3.13 Communication Path

Communication path (Commpath), is the PTP Sync and DelayResp packets path between a master port and a slave port. The slave Commpath state (slaveCommpathState) indicates the current Commpath as been monitored by the slave port.
Up Commpath state (slaveCommpathState=1) indicates that the slave is receiving both Sync packets and DelayResp packets, as expected.
Down Commpath state (slaveCommpathState=0) indicates that the slave is neither receiving Sync packets nor DelayResp Packets. The Commpath is down in case:

- No Sync packets for more than 1 sec
- No DelayResp packets for more than 30 sec.

The Commpath will be down also in case of no Announce packets for more than the TTL period as reported by GetIpcPtpBmcKnownMasterTable (ForeignMaster.expirationCounter). In this case the slave will de-select the master, and as a result the Commpath will be down.
The expirationCounter (TTL) is defined by ptpAnnounceIntervalLog and ptpAnnounceRxTimeout as follow:

- expirationCounter = 2^ptpAnnounceIntervalLog x ptpAnnounceRxTimeout
- IF expirationCounter <2 THEN expirationCounter=2

As an example in G.8275.1 the TTL is 2 sec.

The device state will be changed from trace lock (LK) or (TR), to holdover (HO) or free-run (FR), in case of slaveCommpathState=0.

5.3.13.1 GetIpcPtpCommpathState

Description	Get the Slave's communication path (Commpath) state.			

	`STATUS GetIpcPtpCommpathState(`
	`  IN int32 fnId,`
	`  OUT int32* slaveCommpathState);`			

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603			
Example	rv=GetIpcPtpCommpathState(0,&slaveCommpathState)			
Parameters	Parameter Name		Range	Default
	fnId	0		0
	slaveCommpathState	0	Down	N/A
		1	Up	

5.3.13.2 GetIpcPtpCommpathUpEtime

Description	Get the time elapsed since the last entry to communication path up state measured

	in seconds.
	`STATUS GetIpcPtpCommpathUpEtime(`
	`  IN int32 fnId,`
	`  OUT Uint32* slaveCommpathUpEtime);`

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=GetIpcPtpCommpathUpEtime(0,&slaveCommpathUpEtime)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	slaveCommpathUpEtime	N/A	N/A

5.3.13.3 GetIpcPtpCommpathDnEtime

Description	Get the time elapsed since the last entry to communication path down state measured in seconds.
	`STATUS GetIpcPtpCommpathDnEtime(`
	`  IN int32 fnId,`
	`  OUT Uint32* slaveCommpathDnEtime);`

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=GetIpcPtpCommpathDnEtime(0,&slaveCommpathDnEtime)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	slaveCommpathDnEtime	N/A	N/A

5.3.13.4 GetIpcPtpCommpathStatusCi

Description	Get the current master's communication path status for a slave with channel index CI.
	`STATUS GetIpcPtpCommpathStatusCi(`
	`  IN int32 fnId,`
	`  IN int32 masterCi,`
	`  OUT MasterCommState* masterCommpathState);`

Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode
Example	rv=GetIpcPtpCommpathStatusCi(0,6,&masterCommpathState)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	masterCi	0-63	N/A
	masterCommpathState	N/A	0

5.3.14 Network Performance Monitoring (NPM)

5.3.14.1 SetIpcPtpNpmNetworkRepPeriodicEn

Description	Enable or disable the hourly periodic network report.
	`STATUS SetIpcPtpNpmNetworkRepPeriodicEn(`
	`  IN int32 fnId,`
	`  IN int32 repEn);`

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=SetIpcPtpNpmNetworkRepPeriodicEn(0,1)

Parameters	Parameter Name	Range		Default
	fnId	0		0
	repEn	0	Disable	1
		1	Enable	

5.3.14.2 GetIpcPtpNpmNetworkRepPeriodicEn

Description	Get the hourly periodic network report Enable or Disable status.
	`STATUS GetIpcPtpNpmNetworkRepPeriodicEn(`
	`  IN int32 fnId,`
	`  OUT int32* repEn);`

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=GetIpcPtpNpmNetworkRepPeriodicEn(0,&repEn)

Parameters	Parameter Name	Range	Default

	fnId	0	0
	repEn	See GetIpcPtpNpmNetworkRepPeriodicEn	1

5.3.14.3 GetIpcPtpNpmNetworkRep1min

Description	Get the Slave's network report for an observation time of 1 minute.
	`STATUS GetIpcPtpNpmNetworkRep1min(`
	`  IN int32 fnId,`
	`  OUT NetworkParams* networkParameters,`
	`  OUT NetworkMinMax* parametersMinMax);`

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=GetIpcPtpNpmNetworkRep1min(0,&networkParameters,&parametersMinMax)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	networkParameters	N/A	N/A
	parametersMinMax	N/A	N/A

The report is been updated once per minute and can be used only while the device is in Lock state. i.e. GetIpcPtpState returns state=4.

Since the parametersMinMax is been updated once per minute, the 1 minute report minimal, maximal and average parameters are identical.

5.3.14.4 GetIpcPtpNpmNetworkRep15min

Description	Get the Slave's network report for an observation time of 15 minute.
	`STATUS GetIpcPtpNpmNetworkRep15min(`
	`  IN int32 fnId,`
	`  OUT NetworkParams* networkParameters,`
	`  OUT NetworkMinMax* parametersMinMax);`

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=GetIpcPtpNpmNetworkRep15min(0,&networkParameters,&parametersMinMax)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	networkParameters	N/A	N/A
	parametersMinMax	N/A	N/A

The report is been updated once per minute and can be used only while the device is in Lock state. i.e. GetIpcPtpState returns state=4.

5.3.14.5 GetIpcPtpNpmNetworkRep60min

Description	Get the Slave's network report for an observation time of 60 minute.
	`STATUS GetIpcPtpNpmNetworkRep60min(`
	`  IN int32 fnId,`
	`  OUT NetworkParams* networkParameters,`
	`  OUT NetworkMinMax* parametersMinMax);`

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=GetIpcPtpNpmNetworkRep60min(0,&networkParameters,&parametersMinMax)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	networkParameters	N/A	N/A
	parametersMinMax	N/A	N/A

The report is been updated once per minute and can be used only while the device is in Lock state. i.e. GetIpcPtpState returns state=4.

5.3.14.6 GetIpcPtpNpmNetworkRepInf

Description	Get the Slave's network report since the beginning of starting gathering statistics or since the last time the statistics was reset.
	`STATUS GetIpcPtpNpmNetworkRepInf(`
	`  IN int32 fnId,`
	`  OUT NetworkParams* networkParameters,`
	`  OUT NetworkMinMax* parametersMinMax);`

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603

Example	rv=GetIpcPtpNpmNetworkRepInf(0,&networkParameters,&parametersMinMax)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	networkParameters	N/A	N/A
	parametersMinMax	N/A	N/A

The report is been updated once per minute and can be used only while the device is in Lock state. i.e. GetIpcPtpState returns state=4.

5.3.14.7 SetIpcPtpNpmNetworkRepClr

Description	Reset periodic network report statistics.
	`STATUS SetIpcPtpNpmNetworkRepClr(`
	`  IN int32 fnId);`		

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603		
Example	rv=SetIpcPtpNpmNetworkRepClr(0)		
Parameters	Parameter Name	Range	Default
	fnId	0	0

5.3.14.8 GetIpcPtpNpmPdv

Description	Get the PDV estimation (pdvEst) for the master to slave path in usec during the last minute. This API provides PDV estimation while slave is in: (1) TR state for more than 240 seconds (2) LK state. In case PDV estimation is not available pdvEst=-1.
	`STATUS GetIpcPtpNpmPdv(`
	`  IN int32 fnId,`
	`  OUT int32* pdvEst);`		

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603		
Example	rv=GetIpcPtpNpmPdv(0,&pdvEst)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	pdvEst	0 – 100000 -1 No estimation	0

GetIpcPtpNpmPdv may be not accurate in case device is not in lock or in case of sudden freq change that was not detected yet.

5.3.14.9 SetIpcPtpNpmPeriodicPdvHistEn

Description	Enable or Disable the hourly periodic PDV histogram.
	`STATUS SetIpcPtpNpmPeriodicPdvHistEn(`
	`  IN int32 fnId,`
	`  IN int32 pdvHistEn);`		

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603		
Example	Disabling the periodic PDV Histogram report: rv=SetIpcPtpNpmPeriodicPdvHistEn(0,0)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	pdvHistEn	0 Disable 1 Enable	0

5.3.14.10 GetIpcPtpNpmPeriodicPdvHistEn

Description	Get the hourly periodic PDV histogram Enable or Disable status.
	`STATUS GetIpcPtpNpmPeriodicPdvHistEn(`
	`  IN int32 fnId,`
	`  OUT int32* pdvHistEn);`		

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603		
Example	rv=GetIpcPtpNpmPeriodicPdvHistEn(0,&pdvHistEn)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	pdvHistEn	See SetIpcPtpNpmPeriodicPdvHistEn	0

5.3.14.11 GetIpcPtpNpmPdvHist

Description	Get the PDV histogram statistics.
	`STATUS GetIpcPtpNpmPdvHist(`
	`IN int32 fnId,`
	`IN int32 type,`
	`OUT PdvHist* pdvHist);`

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=GetIpcPtpNpmPdvHist(0,0,&pdvHist)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	type	0	0
	pdvHist	N/A	N/A

Note *This API returns large amount of data and consume significant resources of the management UART (mUART).*

5.3.14.12 SetIpcPtpNpmPdvHistClr

Description	Reset the hourly periodic PDV histogram.
	`STATUS SetIpcPtpNpmPdvHistClr(`
	`IN int32 fnId);`

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=SetIpcPtpNpmPdvHistClr(0)

Parameters	Parameter Name	Range	Default
	fnId	0	0

5.3.14.13 GetIpcPtpNpmFppRep

Description	Get the FPP (Floor Packet Percentage) report.
	`STATUS GetIpcPtpNpmFppRep(`
	`IN int32 fnId,`
	`OUT FppRep* fppRep);`

Applicability	IPC9xxx – BC/Slave modes, IPC17xx Slave mode
Example	rv=GetIpcPtpNpmFppPar(0,&fppRep)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	fppRep	N/A	N/A

5.3.14.14 GetIpcPtpNpmFpp

Description	Get the FPP (Floor Packet Percentage).
	`STATUS GetIpcPtpNpmFpp(`
	`IN int32 fnId,`
	`OUT Fpp* fpp);`

Applicability	IPC9xxx – BC/Slave modes, IPC17xx Slave mode
Example	rv=GetIpcPtpNpmFpp(0,&fpp)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	fpp	N/A	N/A

Note *In case the device is in HO or FR the Percentage and PDV parameters (listed below) shall be zero. Parameters: M2SpercentageN, S2MpercentageN, M2SpdvHistNsMax, S2MpdvHistNsMax.*

5.3.14.15 SetIpcPtpNpmPeriodicFreqEstRep

Description	Enable or Disable the periodic native frequency estimation (Si) report.
	`STATUS SetIpcPtpNpmPeriodicFreqEstRep(`
	`IN int32 fnId,`
	`IN int32 freqEstRepEn);`

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	Disabling the periodic frequency estimation report: rv=SetIpcPtpNpmPeriodicFreqEstRep(0,0)

Parameters	Parameter Name	Range	Default
	fnId	0	0

	freqEstRepEn	0	Disable	0
		1	Enable	

5.3.14.16 GetIpcPtpNpmPeriodicFreqEstRep

Description	Get the periodic native frequency estimation (Si) report Enable or Disable status.
	`STATUS GetIpcPtpNpmPeriodicFreqEstRep(`
	`  IN int32 fnId,`
	`  OUT int32* freqEstRepEn);`

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=GetIpcPtpNpmPeriodicFreqEstRep(0,&freqEstEn)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	freqEstRepEn	See SetIpcPtpNpmPeriodicFreqEstRep	0

5.3.14.17 GetIpcPtpNpmFreqEstRep

Description	Get the native frequency estimation report in ppb for an observation time of 60 minutes.
	`STATUS GetIpcPtpNpmFreqEstRep(`
	`  IN int32 fnId,`
	`  OUT FreqEstRep* freqEstRep);`

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=GetIpcPtpNpmFreqEstRep(0,&freqEstRep)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	freqEstRep	N/A	1

5.4 Administrative API Functions

The administrative API functions set are for configuring controlling and monitoring the device mode of operation.

5.4.1 General Settings

5.4.1.1 SetIpcAdminStatusIndMode

Description	Set the mode of status indication by status pins. Multiplexed mode provides all four possible statuses using three out of the four status pins: STATFR, STATHO and STATLK. In this mode, the STATLK pin is used for indicating both lock (LK) status and trace (TR) status. LK status is indicated by high signal and TR status by an alternate signal. Non-multiplexed mode provides all four possible statuses using four status pins: STATFR, STATHO, STATLK, and STATTR. LK status is indicated by high signal in STATLK pin. TR status is indicated by high signal in STATTR. pin. For more information, refer to IPC9xxx/17xx/1603 datasheet.
	`STATUS SetIpcAdminStatusIndMode(`
	`  IN int32 fnId,`
	`  IN int32 indicationMode);`

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=SetIpcAdminStatusIndMode(0,0)

Parameters	Parameter Name	Range		Default
	fnId	0		0
	indicationMode	0	Multiplexed	1
		1	Non-multiplexed	

5.4.1.2 GetIpcAdminStatusIndMode

Description	Get the mode of status indication by status pins.
	`STATUS GetIpcAdminStatusIndMode(`
	`  IN int32 fnId,`
	`  OUT int32* indicationMode);`

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=GetIpcAdminStatusIndMode(0,&indicationMode)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	indicationMode	See SetIpcAdminStatusIndMode	1

5.4.1.3 SetIpcAdminMasterSqMode

Description	While master squelch is in Auto mode (masterSqMode=1) the master send event messages, only in case TOD is stable. All other packets are been handled by the master without any change (e.g. Announce, RUTS, GRUTS).
	`STATUS SetIpcAdminMasterSqMode(`
	`  IN int32 fnId,`
	`  IN int32 masterSqMode);`

Applicability	IPC9xxx – Master mode, IPC17xx – Master mode
Example	rv=SetIpcAdminMasterSqMode(0,0)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	masterSqMode	0 Disable 1 Auto	0

ToD stable is defined as consecutive 30-40 sec (sqTime2), without ToD updates.

5.4.1.4 GetIpcAdminMasterSqMode

Description	Get the master squelch mode.
	`STATUS GetIpcAdminMasterSqMode(`
	`  IN int32 fnId,`
	`  OUT int32* masterSqMode);`

Applicability	IPC9xxx – Master mode, IPC17xx – Master mode
Example	rv=GetIpcAdminMasterSqMode(0,&masterSqMode)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	masterSqMode	See SetIpcAdminMasterSqMode	

5.4.1.5 GetIpcAdminMasterSqState

Description	Get the master squelch state.
	`STATUS GetIpcAdminMasterSqState(`
	`  IN int32 fnId,`
	`  OUT int32* masterSqState);`

Applicability	IPC9xxx – Master mode, IPC17xx – Master mode
Example	rv=GetIpcAdminMasterSqState(0,&masterSqState)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	masterSqState	0 Master in squelch. Master doesn't send event messages 1 Master doesn't in squelch. Master does send event messages	

5.4.1.6 SetIpcAdminRelock

Description	Relock the Slave to the parent master. This API can be used by the user in order to shorten the slave relock time. For optimal performance it's recomanded that the slave will be in slaveStateHoReady > 0 as reported by GetIpcPtpStateSlaveHoReady.
	`STATUS SetIpcAdminRelock(`
	`  IN int32 fnId,`
	`  IN int32 mode);`

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=SetIpcAdminRelock(0,0)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	mode	0 – Relock - TOD was changed 5 – Relock - SyncE PLL moved to LOCK	0

	6 – Relock - SyncE PLL moved to FREERUN

5.4.1.7 SetIpcAdminPpsOutAsymmetryCorr

Description	Set the 1PPS output asymmetry correction phase (phase offset) in nsec. This API changes the device ToD.

```
STATUS SetIpcAdminPpsOutAsymmetryCorr(
    IN int32 fnId,
    IN int32 corrAsymmetry);
```

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=SetIpcAdminPpsOutAsymmetryCorr(0,100)

Parameters

Parameter Name	Range	Default
fnId	0	0
corrAsymmetry	-100000 – +100000	0

The SetIpcAdminPpsOutAsymmetryCorr better be used while the slave is in free run, before slave move into Trace. In case it's not, then setting of corrAsymmetry may influence the 1PPS output with a delay of several minutes.

5.4.1.8 GetIpcAdminPpsOutAsymmetryCorr

Description	Get the 1PPS output asymmetry correction phase (phase offset) in nsec.

```
STATUS GetIpcAdminPpsOutAsymmetryCorr(
    IN int32 fnId,
    OUT int32* corrAsymmetry);
```

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=GetIpcAdminPpsOutAsymmetryCorr(0,&corrAsymmetry)

Parameters

Parameter Name	Range	Default
fnId	0	0
corrAsymmetry	See SetIpcAdminPpsOutAsymmetryCorr	0

5.4.1.9 SetIpcAdminPpsOutMode

Description	Set the PPSOUT mode.

```
STATUS SetIpcAdminPpsOutMode(
    IN int32 fnId,
    IN int32 phaseOutMode);
```

Applicability	IPC9xxx, IPC17xx
Example	rv=SetIpcAdminPpsOutMode(0,1)

Parameters

Parameter Name		Range	Default
fnId	0		0
ppsOutMode	0	Recovered 1PPS	1
	1	Recovered 1PPS with phase offset	
	5	Three state	

SetIpcAdminPpsOutPhaseOfs shall be used in order to set the phase offset while ppsOutMode=1.

5.4.1.10 GetIpcAdminPpsOutMode

Description	Get the PPSOUT mode.

```
STATUS GetIpcAdminPpsOutMode(
    IN int32 fnId,
    OUT int32* ppsOutMode);
```

Applicability	IPC9xxx, IPC17xx
Example	rv=GetIpcAdminPpsOutMode(0,&ppsOutMode)

Parameters

Parameter Name	Range	Default
fnId	0	0
ppsOutMode	See SetIpcAdminPpsOutMode	0

5.4.1.11 SetIpcAdminPpsOutPhaseOfs

Description	Set the 1PPS output phase offset in nsec. This API does not change the device ToD.

	STATUS SetIpcAdminPpsOutPhaseOfs(**IN int32 fnId,** **IN int32 phaseOfs);**
Applicability	IPC9xxx, IPC17xx
Example	rv=SetIpcAdminPpsOutPhaseOfs(0,100)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	phaseOfs	-100000 – +100000	0

For advancing the 1PPS output vs. device time use phaseOfs<0.
For delaying the 1PPS output vs. device time use phaseOfs>0.

5.4.1.12 GetIpcAdminPpsOutPhaseOfs

Description	Get the 1PPS output phase offset in nsec.
	STATUS GetIpcAdminPpsOutPhaseOfs(**IN int32 fnId,** **OUT int32* phaseOfs);**
Applicability	IPC9xxx, IPC17xx
Example	rv=GetIpcAdminPpsOutPhaseOfs(0,&phaseOfs)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	phaseOfs	See SetIpcAdminPpsOutPhaseOfs	0

5.4.1.13 SetIpcAdminPpsInPhaseOfs

Description	Set the 1PPS input's phase offset in nsec.
	STATUS SetIpcAdminPpsInPhaseOfs(**IN int32 fnId,** **IN int32 phaseOfs);**
Applicability	IPC9xxx – Master, IPC17xx – Master
Example	rv=SetIpcAdminPpsInPhaseOfs(0,100)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	phaseOfs	-100000 – +100000	0

The SetIpcAdminPpsInPhaseOfs better be used while the master is in free run, before master move into Trace. In case it's not, then setting of phaseOfs may influence the 1PPS input phase with a delay of several minutes.

For advancing the device time vs. 1PPS input use phaseOfs>0.
For delaying the device time vs. 1PPS input use phaseOfs<0.

5.4.1.14 GetIpcAdminPpsInPhaseOfs

Description	Get the 1PPS input's phase offset in nsec.
	STATUS GetIpcAdminPpsInPhaseOfs(**IN int32 fnId,** **OUT int32* phaseOfs);**
Applicability	IPC9xxx – Master, IPC17xx – Master
Example	rv=GetIpcAdminPpsInPhaseOfs(0,&phaseOfs)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	phaseOfs	See SetIpcAdminPpsInPhaseOfs	0

5.4.1.15 SetIpcAdminSrvMode

Description	Set the servo mode. In order to comply with G.8273.2 noise transfer, the servo shall be set to srvMode=2. In this mode the servo performance, in the present of high PDV, e.g. G.8261 cases, will be degraded.
	STATUS SetIpcAdminSrvMode(**IN int32 fnId,** **IN int32 SrvMode);**
Applicability	IPC9xxx – BC/Slave modes

Example	rv=SetIpcAdminSrvMode(0,2)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	srvMode	0 – Optimal PDV rejection	0
		1 – Reserved	
		2 – Optimal noise transfer for full timing support network	

5.4.1.16 GetIpcAdminSrvMode

Description	Get the servo mode. **STATUS GetIpcAdminSrvMode(** **IN int32 fnId,** **OUT int32* SrvMode);**		
Applicability	IPC9xxx – BC/Slave mode		
Example	rv=GetIpcAdminSrvMode(0,&srvMode)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	srvModePar	See SetIpcAdminSrvMode	N/A

5.4.1.17 SetIpcAdminIpDscp

Description	Set the IPv4 DSCP field. IpDscpPtpEvent set the DSCP field of PTP event packets (UDP port 319). IpDscpPtpGeneral set the DSCP field of PTP general packets (UDP port 320). **STATUS SetIpcAdminIpDscp(** **IN int32 fnId,** **IN u_char ipDscpPtpEvent,** **IN u_char ipDscpPtpGeneral);**		
Applicability	IPC9xxx, IPC17xx		
Example	rv=SetIpcAdminIpDscp(0,59,47)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	ipDscpPtpEvent	0 – 63	59
	ipDscpPtpGeneral	0 – 63	47

5.4.1.18 GetIpcAdminIpDscp

Description	Get the IPv4 DSCP field. **STATUS GetIpcAdminIpDscp(** **IN int32 fnId,** **OUT IpDscp* ipDscp);**		
Applicability	IPC9xxx, IPC17xx		
Example	rv=GetIpcAdminIpDscp(0,&ipDscp)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	ipDscp	See SetIpcAdminIpDscp	

5.4.1.19 SetIpcAdminIpDsf

Description	Set the IPv4 Differential Services Fields (DSCP & ECN). IpDscpPtpEvent & IpEcnPtpEvent set the DSCP & ECN fields of PTP event packets (UDP port 319). IpDscpPtpGeneral & IpEcnPtpGeneral set the DSCP & ECN field of PTP general packets (UDP port 320). **STATUS SetIpcAdminIpDsf(** **IN int32 fnId,** **IN u_char ipDscpPtpEvent,** **IN u_char ipEcnPtpEvent,** **IN u_char ipDscpPtpGeneral,** **IN u_char ipEcnPtpGeneral);**		
Applicability	IPC9xxx, IPC17xx		
Example	rv=SetIpcAdminIpDsf(0,59,0,47,0)		

Parameters	Parameter Name	Range	Default
	fnId	0	0
	ipDscpPtpEvent	0 – 63	59
	ipEcnPtpEvent	0 – 3	0
	ipDscpPtpGeneral	0 – 63	47
	ipEcnPtpGeneral	0 – 3	0

5.4.1.20 GetIpcAdminIpDsf

Description	Get the IPv4 Differential Services Fields (DSCP & ECN).
	`STATUS GetIpcAdminIpDsf(`
	`  IN int32 fnId,`
	`  OUT IpDsf* ipDsf);`

Applicability	IPC9xxx, IPC17xx
Example	rv=GetIpcAdminIpDsf(0,&ipDsf)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	ipDsf	See SetIpcAdminIpDsf	

5.4.2 Channels Allocation & Reports

5.4.2.1 General

The master responsible for channels allocation to the slaves. This chapter describes the limitations per packet type.

5.4.2.1.1 Announce Channel Allocation

The transmitted Announce packet allocation is performed in accordance with the following limits:

- Lower limit per channel – 1/16pps (logInterMessagePeriod=4)
- Upper limit per channel – 8pps (logInterMessagePeriod=-3)
- Aggregated limits – See “Aggregated Unified Packet Rate Mode” chapter.

In case the requested packet rate as received by the RUTS (REQUEST_UNICAST_TRANSMISSION) packets exceed the per channel limits, the master shall allocate Announce channel with packet rate up to the limit pending available resources. The master shall respond with GRUTS (GRANT_UNICAST_TRANSMISSION) with proper logInterMessagePeriod.

5.4.2.1.2 Sync Channel Allocation

The transmitted Sync / FU packet allocation is performed in accordance with the following limits:

- Lower limit per channel – 1/16pps (logInterMessagePeriod=4)
- Upper limit per channel, Master – 128pps (logInterMessagePeriod=-7)
- Upper limit per channel, BC-Master – 32pps (logInterMessagePeriod=-5)
- Aggregated limits – See “Aggregated Unified Packet Rate Mode” chapter.

In case the requested packet rate as received by the RUTS (*REQUEST*_UNICAST_TRANSMISSION) packets exceed the per channel limits, the master shall not allocate Sync channel. The master shall respond with DGRUTS (GRANT_UNICAST_TRANSMISSION with durationField 0 == Denial).
In addition, the following message shall be reported to the local device log – “Cannot allocate new ch (4,X)” while X is the requested packet rate.

Notes:

- The slave lower Sync / FU limit is 2pps. (logInterMessagePeriod=-1).
- The recommended upper limit of 32pps for BC Master port, can be disabled by bcPktRateLimitMode controled by SetIpcAdminAutoSlaveAddRelPar. In case of disabling that limit, part of the DelayReq may not be replied with DelayResp. The number of DelayReq packets that were not replied with DelayResp (i.e. dropped) is reported by GetIpcPtpCntRepMaster.
- In case PTP-Slave sends RUTS (REQUEST_UNICAST_TRANSMISSION) within limits and channel was allocated, and later PTP-Slave request packet rate that exceed per channel limits, The master immediately sends DGRUTS (GRANT_UNICAST_TRANSMISSION with durationField 0 == Denial), and stops transmitting message after duration time expires.

5.4.2.1.3 DelayResp Channel Allocation

The DelayReq / DelayResp packet allocation is performed in accordance with the following limits:

- Lower limit per channel – 1/16pps (logInterMessagePeriod=4)
- Upper limit per channel, Master – 128pps (logInterMessagePeriod=-7)
- Upper limit per channel, BC-Master – 32pps (logInterMessagePeriod=-5)
- Aggregated limits – See “Aggregated Unified Packet Rate Mode” chapter.

In case the requested packet rate as received by the RUTS (REQUEST_UNICAST_TRANSMISSION) packets exceed the per channel limits, the master shall not allocate Sync channel. The master shall respond with DGRUTS (GRANT_UNICAST_TRANSMISSION with durationField 0 == Denial).
In addition, the following message shall be reported to the local device log – “Cannot allocate new ch (5,X)” while X is the requested packet rate.

Notes:

- The recommended upper limit of 32pps for BC Master port, can be disabled by bcPktRateLimitMode controled by SetIpcAdminAutoSlaveAddRelPar. In case of disabling that limit, part of the DelayReq may not be replied with DelayResp. The number of DelayReq packets that were not replied with DelayResp (i.e. dropped) is reported by GetIpcPtpCntRepMaster.
- In case PTP-Slave sends RUTS (REQUEST_UNICAST_TRANSMISSION) within limits and channel was allocated, and later PTP-Slave request packet rate that exceed per channel limits, The master immediately sends DGRUTS (GRANT_UNICAST_TRANSMISSION with durationField 0 == Denial), and stops transmitting message after duration time expires.
- The master expect that the a slave will send DelayReq in accordance with the GRUTS. It doesn't perform any specific action in case of packet rate violation (i.e. drop part of the packets).
- The master supports DelayReq packet rate of up to 128 pps per channel. However for high performance, the recommended DelayReq packet rate range is 8-32pkt/sec. Higher packet rate, doesn't improve the PTP synchronization performance of the slave.
- The master port may not respond to all the DelayReq packets with DelayResp while packet rate exceed 32 pps.

5.4.2.1.4 Aggregated Unified Packet Rate Mode

5.4.2.1.4.1 General

The IPC9xxx and IPC17xx master ports support up to 64 slaves/channels. The unified packets are been grouped into two types:
Group-1: Sync, FU, Delay Request and Delay Response
Group-2: Announce
Each group has separate resources and allocation mechanism.

5.4.2.1.4.2 Group-1: Sync, FU, Delay Request and Response

In Master, the Maximal Aggregated Unified Packet Rate (MAUPR) is 2000pps (packet per sec) for IPC9xxx/IPC17xx in unicast mode, 1400pps for IPC9xxx/IPC17xx in multicast mode, 1400pps for IPC9xxx in BC mode, and 1300pps for IPC1703. The Aggregated Unified Packet Rate (AUPR) includes the transmitted Sync, FU (in case of two steps), Delay response and received Delay request packets.

As an example, in case of IPC9xxx in unicast Master mode with 32pps Sync per slave, one step, 8pps Delay response, the AUPR shall be 48pps (32+8+8=48) for a single slave, 768pps (48x16=768) for 16 slaves, 1536pps (48x32=1536) for 32 slaves.

The MAUPR is always guaranteed. In order to maximize the device utilization, the allocation mechanism may allocate several hundreds of packets per second beyond the MAUPR. The specific extra allocation can be up to 256pps and varied according to the device operation conditions.

The aggregated unified packet rate modes (AUPRMode) are:

- Unlimited – Master allocates unlimited service for up to 64 slaves. The MAUPR is not applicable.
- Adaptive – Master allocates service up to 64 slaves, however the AUPR shall be kept below or equal the MAUPR by reducing the Sync and FU (in case of two steps) packet rate by a factor (PreScale) for all the slaves if required.

- Limited – Master allocates service up to 64 slaves, however the AUPR shall be kept below or equal the MAUPR by rejecting service requested by slaves in case it pass the MAUPR. In case of exceeding MAUPR, while startMode=22,24, Master allocates to each channel: Max(logSyncInterval,-floor(log2(MAUPR/N))). N is define as follow:
  - In case portMapMode=0: N is the number of active channels
  - In case portMapMode!=0: N is the AcceptableTableAll.actualTableSize

5.4.2.1.4.3 Group-2: Announce

In unicast Master and BC (startMode 0 and 4 respectively), the Maximal Aggregated Unified Packet Rate (MAUPR) is 128pps (packet per sec) for IPC9xxx in BC/Slave/Master modes, and IPC17xx in Slave/Master modes. The MAUPR and for IPC1703 is 32pps. The Aggregated Unified Packet Rate (AUPR) includes the transmitted Announce packets. The AUPR includes the Announce packets per sec from all channels. Channels with Announce packet rate below 1pps, are negligible and counted as 0pps for the AUPR.

As an example, in case of IPC9xxx in unicast Master mode with 1pps Announce per slave, the AUPR shall be 1pps for a single slave, 16pps (1x16=32) for 16 slaves, 64pps (1x64=64) for 64 slaves.

The aggregated Unified packet rate modes (AUPRMode) are:

- Unlimited – Master allocates unlimited service for up to 64 slaves. The MAUPR is not applicable. The actual transmitted Announce packets is define by ptpAnnounceMode (Set by SetIpcPtpAnnounceTxMode) and ptpAnnounceIntervalLog (Set by SetIpcPtpAnnounceIntervalLog).
- Limited – Master allocates Announce service up to MAUPR, and up to 64 slaves. The AUPR shall be kept below or equal the MAUPR. In case of exceeding MAUPR, while ptpAnnounceMode=2,3,4, new service requested will be limited to maximal packet rate allocation of 1 Announce packet per 2sec. In case of exceeding MAUPR, while ptpAnnounceMode=1, Master allocates to each channel: Max(ptpAnnounceIntervalLog,-floor(log2(MAUPR/N))). While N defined as follow:
  - In case of startMode=0,2,4, N is define as the number of active channels.
  - In case of startMode=22,24 and portMapMode=0, N is define as the number of active channels.
  - In case of startMode=22,24 and portMapMode!=0, N is define as the AcceptableTableAll.actualTableSize.

5.4.2.2 GetIpcPtpAnnounceTable

Description	Get the Master's AnnounceTable.
	`STATUS GetIpcPtpAnnounceTable(`
	`  IN int32 fnId,`
	`  OUT AnnounceTable* announceTable);`

Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode
Example	rv=GetIpcPtpAnnounceTable(0,&announceTable)

Parameters

Parameter Name	Range	Default
fnId	0	0
announceTable	N/A	N/A

This API returns large amount of data and consume significant resources of the management UART (mUART).

5.4.2.3 SetIpcPtpAnnounceTxMode

Description	Set the Announce packet transmission mode.
	`STATUS SetIpcPtpAnnounceTxMode(`
	`  IN int32 fnId,`
	`  IN int32 ptpAnnounceMode);`

Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode
Example	rv=SetIpcPtpAnnounceTxMode(0,1)

Parameters

Parameter Name	Range		Default
fnId	0		0
ptpAnnounceMode	0	Disable	1,2
	1	Auto-1 – packet rate sets by master ptpAnnounceIntervalLog	

2	Auto-2 – packet rate sets by slave request, as record by grantRutsAnnounce table, pending available resource.
3	Auto-3 – packet rate sets by slave request, as record by grantRutsAnnounce table up to ptpAnnounceIntervalLog, pending available resource.
4	Manual Announce channel allocation – new slaves shall be added to AnnounceTable by the user using SetIpcPtpAdminAnnounceAdd API. slaves shall be released from AnnounceTable by the user using SetIpcPtpAdminAnnounceDelete / SetIpcPtpAdminAnnounceDeleteAll API. Aging mechanism is disabled.

Note *In Auto modes (1,2,3), new slaves shall be added to AnnounceTable following automatic allocation of master resource driven by accepting RUTS packet for Announce. Slaves shall be released from AnnounceTable using aging mechanism or by the user using SetIpcPtpAdminAnnounceDelete / SetIpcPtpAdminAnnounceDeleteAll API.*

Note *Manual Announce channel allocation (4) applicable for startMode = 0,4 only.*
In this mode CANCEL_UNICAST_TRANSMISSION doesn't cause deletion of the Announce channel from the announceTable.

Note *Default:*
ptpAnnounceMode=1: startMode = 2/7/22/24 (all multicast master/BC modes)
ptpAnnounceMode=2: startMode = 0,4 (all unicast master/BC modes)

5.4.2.4 GetIpcPtpAnnounceTxMode

Description	Get the Announce packet transmission mode.
	`STATUS GetIpcPtpAnnounceTxMode(`
	`  IN int32 fnId,`
	`  OUT int32* ptpAnnounceMode);`

Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode
Example	rv=GetIpcPtpAnnounceTxMode(0,&ptpAnnounceMode)

Parameters

Parameter Name	Range	Default
fnId	0	0
ptpAnnounceMode	See SetIpcPtpAnnounceTxMode	

5.4.2.5 SetIpcPtpAnnounceIntervalLog

Description	Set the ptpAnnounceIntervalLog of the master port.
	`STATUS SetIpcPtpAnnounceIntervalLog(`
	`  IN int32 fnId,`
	`  IN Uint16 portIndex,`
	`  IN Uint32 ptpAnnounceIntervalLog);`

Applicability	portMapMode=0: IPC9xxx – BC/Master modes, IPC17xx – Master mode portMapMode=>1: IPC9xxx – BC/Master modes
Example	rv=SetIpcPtpAnnounceIntervalLog(0,2,4)

Parameters

Parameter Name	Range	Default
fnId	0	0
portIndex	portMapMode=0 2 BC-Master port 2 Master mode portMapMode=>1 0 All ports 1-N Specific port	N/A
ptpAnnounceIntervalLog	-3 .. 4	1 (profile-0,1) -3 (profile-2)

5.4.2.6 GetIpcPtpAnnounceIntervalLog

Description	Get the ptpAnnounceIntervalLog. For Master port, returns the preconfigured ptpAnnounceIntervalLog.
	`STATUS GetIpcPtpAnnounceInterval(`
	`IN int32 fnId,`
	`IN Uint16 portIndex,`
	`OUT Uint32* ptpAnnounceIntervalLog);`

Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode
Example	rv=GetIpcPtpAnnounceIntervalLog(0,2,&ptpAnnounceIntervalLog)

Parameters

Parameter Name	Range	Default
fnId	0	0
portIndex	See SetIpcPtpAnnounceInterval	
ptpAnnounceIntervalLog	See SetIpcPtpAnnounceInterval	

5.4.2.7 SetIpcPtpAdminAnnounceAdd

Description	Add a unicast Master to AnnounceTable. This API is applicable only for ptpAnnounceMode=4.
	`STATUS SetIpcPtpAdminAnnounceAdd(`
	`IN int32 fnId,`
	`IN char* portAddress,`
	`IN char* portIdentity,`
	`IN Int16 logPacketPeriod,`
	`IN Uint16 reserved);`

Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode
Example	rv=SetIpcPtpAdminAnnounceAdd(0,"192.168.100.10" ,"00:00:00:00:00:00:00:00:0000",1,2)

Parameters

Parameter Name	Range	Default
fnId	0	0
portAddress	"0.0.0.0" – "255.255.255.255"	N/A
portIdentity	"00:00:00:00:00:00:00:00:0000 - FF:FF:FF:FF:FF:FF:FF:FF:FFFF"	N/A
logPacketPeriod	-3.. 4	N/A
reserved	2	N/A

Note *portIdentity is optional field. It does not use for Announce packet generation. It uses only for local display of the grantRutsAnnounce.*

Note *The reserved field shall be set to 2.*

5.4.2.8 SetIpcPtpAdminAnnounceDelete

Description	Delete a unicast Master from AnnounceTable.
	`STATUS SetIpcPtpAdminAnnounceDelete(`
	`IN int32 fnId,`
	`IN char* portAddress);`

Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode
Example	rv=SetIpcPtpAdminAnnounceDelete(0,"192.168.100.10")

Parameters

Parameter Name	Range	Default
fnId	0	0
portAddress	"0.0.0.0" – "255.255.255.255"	N/A

Note *The Master stop sending Announce messages to the deleted Slave. The Master send CANCEL_UNICAST_TRANSMISSION(An) message to the Slave.*

5.4.2.9 SetIpcPtpAdminAnnounceDeleteAll

Description	Delete all unicast Masters from AnnounceTable.
	`STATUS SetIpcPtpAdminAnnounceDeleteAll(`
	`IN int32 fnId);`

Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode

Example	rv=SetIpcPtpAdminAnnounceDeleteAll(0)		
Parameters	Parameter Name	Range	Default
	fnId	0	0

The Master stop sending Announce messages to all the Slaves. The Master send CANCEL_UNICAST_TRANSMISSION(An) message to all Slaves.

5.4.2.10 SetIpcPtpAdminSlaveServedMode

Description: Set the SlaveServedMode.

```
STATUS SetIpcPtpAdminSlaveServedMode(
    IN int32 fnId,
    IN int32 slaveServedMode);
```

Applicability: IPC9xxx – BC/Master modes, IPC17xx – Master mode

Example: rv=SetIpcPtpAdminSlaveServedMode(0,1)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	slaveServedMode	0 – Present slave MAC address	0 - ptpPortMap = 0
		1 – Present slave MAC based on portIdentity (applicable for srcMAC mapping). clockIdentity shall be in accordance with IEEE EUI-48 as per IEEE 1588 standard.	1 - ptpPortMap !=0

5.4.2.11 GetIpcPtpAdminSlaveServedMode

Description: Get the slaveServedMode.

```
STATUS GetIpcPtpAdminSlaveServedMode(
    IN int32 fnId,
    OUT int32* slaveServedMode);
```

Applicability: IPC9xxx – BC/Master modes, IPC17xx – Master mode

Example: rv=GetIpcPtpAdminSlaveServedMode(0,&slaveServedMode)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	slaveServedMode	See SetIpcPtpAdminSlaveServedMode	

5.4.2.12 GetIpcPtpAdminSlavesServed

Description: Get the report of Slaves being served with Sync and DelayResp services by the Master.

```
STATUS GetIpcPtpAdminSlavesServed(
    IN int32 fnId,
    IN int32 page,
    OUT SlavesServed* slavesServed);
```

Applicability: IPC9xxx – BC/Master modes, IPC17xx – Master mode

Example: rv=GetIpcPtpAdminSlavesServed(0,1,&slavesServed)

Parameters	Parameter Name	Range		Default
	fnId	0		0
	page	0	CI 0-15	0
		1	CI 16-31	
		2	CI 32-47	
		3	CI 48-63	
	slavesServed	N/A		N/A

STATUS=-7 in case of empty page. (no slaves within that page)

5.4.2.13 SetIpcPtpAdminAutoSlaveAddRelMode

Description: Control the adding or releasing services for slaves.

Add service:

ncaMode=3 (startMode=2,7,22,24): The master port automatically add a slave to

	the slavesServed table. The slavesServed table contains the Sync (single multicast service at channel 0) and DelayResp services. The process of adding a slave starts by: a. startMode=7,22,24: receiving DelayReq packet. Depending on resources availability, the master would either grant or deny the request. b. startMode=2: receiving Request Unicast Transmission Signaling (RUTS, REQUEST UNICAST TRANSMISSION) packet for DelayResp. Depending on resources availability, the master would either grant or deny the request. If granted, the master shall send the slave Grant RUTS packet. If denied, the master shall send the slave Denial RUTS packet. *ncaMode=5 (startMode=0,4):* The master port automatically add a slave to the slavesServed table. The slavesServed table contains the Sync and DelayResp services. The process of adding a slave starts by receiving Request Unicast Transmission Signaling (RUTS, REQUEST UNICAST TRANSMISSION) packet for Sync or DelayResp. Depending on resources availability, the master would either grant or deny the request. If granted, the master shall send the slave Grant RUTS packet. If denied, the master shall send the slave Denial RUTS packet. **Resolution of service period:** *ncaMode=3 (startMode=2,7,22,24):* The master port ignore the durationField of RUTS packet for Sync and DelayResp and use a local configurable parameter, ncaAgeingTh, set by SetIpcPtpAdminAutoSlaveAddRelAgeTh. *ncaMode=5 (startMode=0,4):* The master port uses the durationField of RUTS packet for Sync & DelayResp. In case durationField is in the range of 10 to 1000 sec, it will be used. In case durationField is below 10 sec, 10 sec will be used. In case durationField is above 1000 sec, 1000 sec will be used. **Aging:** *ncaMode=3 (startMode=2,7,22,24):* The DelayResp channel shall be aged (i.e. removed from the slavesServed table as been reported by GetIpcPtpAdminSlavesServed) upon: a. startMode=7,22,24: reaching expiration of ncaAgeingTh due to not receiving DelayReq by the slave b. startMode=2: reaching expiration of ncaAgeingTh due to not receiving RUTS for DelayResp by the slave or upon receiving CANCEL_UNICAST_TRANSMISSION TLV for DelayResp. *ncaMode=5 (startMode=0,4):* The Sync channel shall be aged (i.e. removed from the slavesServed table as been reported by GetIpcPtpAdminSlavesServed) upon reaching expiration of durationField of Sync (MasterChannelReport.timeToLiveSy) or upon receiving CANCEL_UNICAST_TRANSMISSION TLV for Sync. The DelayResp service will be no longer available upon reaching expiration of durationField of DelayResp (MasterChannelReport.timeToLiveDr) or upon receiving CANCEL_UNICAST_TRANSMISSION TLV for DelayResp. **STATUS SetIpcPtpAdminAutoSlaveAddRelMode(** **IN int32 fnId,** **IN int32 ncaMode);**
Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode
Example	rv=SetIpcPtpAdminAutoSlaveAddRelMode(0,3)

Parameters

Parameter Name	Range	Default
fnId	0	0
ncaMode	0 Manual Slave add/release 3 Auto Slave add/release - use ncaAgeingTh 5 Auto Slave add/release - use durationField	5:startMode=0,4 3:startMode=2,7,22,24

5.4.2.14 GetIpcPtpAdminAutoSlaveAddRelMode

Description	Get Slave add mode. **STATUS GetIpcPtpAdminAutoSlaveAddRelMode(** **IN int32 fnId,** **OUT int32* ncaMode);**
Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode

Example	rv=GetIpcPtpAdminAutoSlaveAddRelMode(0,&ncaMode)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	ncaMode	See SetIpcPtpAdminAutoSlaveAddRelMode	

5.4.2.15 SetIpcPtpAdminAutoSlaveAddRelAgeTh

Description	Set the ageing threshold in seconds. In case ncaMode=1,3 the Master discontinue sending sync packets to a Slave when DelayReq or RUTS (REQUEST UNICAST TRANSMISSION TLV) were not received from the slave by the Master within ncaAgeingTh or 60sec, the earlier. The channel is aged (i.e. removed from the slavesServed table as been reported by GetIpcPtpAdminSlavesServed) upon reaching ncaAgeingTh or upon receiving CANCEL_UNICAST_TRANSMISSION TLV. Ageing time is measured in seconds. It is activated if the Commpath state of a specific channel is in DOWN state.

```
STATUS SetIpcPtpAdminAutoSlaveAddRelAgeTh(
    IN int32 fnId,
    IN Uint32 ncaAgeingTh);
```

Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode		
Example	Setting ageing time to 1800sec (30min): rv=SetIpcPtpAdminAutoSlaveAddRelAgeTh(0,1800)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	ncaAgeingTh	10 – 3600	10/300

Note *ncaAgeingTh can be set to values in the range of 10 – 3600. However, ncaAgeingTh below 30 may cause suboptimal operation and is not recommended.*

Note *Default:*
StartMode=22,24 ncaAgeingTh=10
StartMode=all other ncaAgeingTh=300

5.4.2.16 GetIpcPtpAdminAutoSlaveAddRelAgeTh

Description	Get auto add/release Slave ageing threshold.

```
STATUS GetIpcPtpAdminAutoSlaveAddRelAgeTh(
    IN int32 fnId,
    OUT Uint32* ncaAgeingTh);
```

Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode		
Example	rv=GetIpcPtpAdminAutoSlaveAddRelAgeTh(0,&ncaAgeingTh)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	ncaAgeingTh	See SetIpcPtpAdminAutoSlaveAddRelAgeTh	

5.4.2.17 SetIpcAdminAutoSlaveAddRelPar

Description	Set the ncaPar.

```
STATUS SetIpcAdminAutoSlaveAddRelPar(
    IN int32 fnId,
    IN int32 bcPktRateLimitMode,
    IN int32 limit4x128ch,
    IN int32 addDReq1);
```

Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode			
Example	rv=SetIpcAdminAutoSlaveAddRelPar(0,0,0,0)			
Parameters	Parameter Name		Range	Default
	fnId	0		0
	bcPktRateLimitMode	0	Limited	0
		1	Unlimited	
	limit4x128ch	0	Limited	0
		1	Unlimited	
	addDReq1	0	Limited	0
		1	Unlimited	

bcPktRateLimitMode - Set the mode of BC Packet Rate Limitation.
In BcPktRateLimitMode=0:
(1) The maximal Sync packet rate (pktRate) of the BC Slave port, as sets by SetIpcPtpAdminStart is 32pps.
(2) The maximal Sync packet rate allocated by the BC Master port while ncaMode=3,5 is 32pps. Higher requested Sync packet rate shall be rejected. As a result, the channel_Type_ *shall be greater or equal to 3.*

Note *In bcPktRateLimitMode=1: no special limitations in BC mode.*

In case bcPktRateLimitMode=0 and the maximal Sync packet rate (pktRate) of the BC Slave port, as sets by SetIpcPtpAdminStart exceed 32pps, the following message shall be present:
[#0201] Sync packet rate (pktRate) exceeds max rate - pktRate set to max rate.

In case bcPktRateLimitMode=0 and the requested Sync packet rate by other slaves, from the BC Master port, exceed 32pps, the following message shall be present:
[#0201] Cannot allocate new ch (3).

limit4x128ch – Set the limit of maximal 4 channels of 128pps Sync. It is recommended not to exceed maximal number of 4 Sync channels of 128pps. The recommended upper limit of 4 channels of 128pps for startMode=0,4, can be disabled by limit4x128ch controlled by SetIpcAdminAutoSlaveAddRelPar. Additional requests for 128pps sync channels shall be rejected. In case of disabling that limit, the master may not operate as required.

Note *The limit4x128ch is applicable for startMode=4 only in case bcPktRateLimitMode=1. In case bcPktRateLimitMode=0, even a single 128pps Sync channel cannot be allocated.*

In case limit4x128ch=0 and the requested Sync packet rate by other slaves, exceed 4 channels of 128pps Sync, the following message shall be present:
[#0201] Cannot allocate new ch (6).

Note *addDReq1 – In order to verify robust allocation of new Sync channels, the master port considers slaves able to request up to ptpDelayReqInterval=1. This mode is applicable for startMode=0,4. The recommended allocation considering slaves able to request up to ptpDelayReqInterval=1 can be disabled by addDReq1 controlled by SetIpcAdminAutoSlaveAddRelPar*

Note *In order to change the device settings, the following sequence shall be performed, in all devices, after invoking the API:*
SetIpcAdminStore
SetIpcPtpAdminStop
SetIpcPtpAdminStart
Or
SetIpcAdminStore
SetIpcPtpAdminReset

5.4.2.18 GetIpcAdminAutoSlaveAddRelPar

Description	Get the ncaPar.
	`STATUS GetIpcAdminAutoSlaveAddRelPar(`
	`  IN int32 fnId,`
	`  OUT NcaPar* ncaPar);`

Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode
Example	rv=GetIpcAdminAutoSlaveAddRelPar(0,&ncaPar)
Parameters	Parameter Name / Range / Default fnId / 0 / 0 ncaPar / See SetIpcAdminAutoSlaveAddRelPar

5.4.2.19 SetIpcPtpAdminManSlaveAddCi

Description	Manually assign a Slave by allocating Channel Index (CI) includes Sync and DelayReq / DelayResp services. The Master commence sending unicast sync packets to the Slave and respond to Delay Request received from that Slave.
	`STATUS SetIpcPtpAdminManSlaveAddCi(`
	`  IN int32 fnId,`
	`  IN Uint32 CI,`
	`  IN char* slaveAddress,`
	`  IN int32 logSyncInterval);`

Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode

Example	32 Sync pkt/sec: rv=SetIpcPtpAdminManSlaveAddCi(0,5,"192.168.1.107",-5) 128 Sync pkt/sec: rv=SetIpcPtpAdminManSlaveAddCi(0,8,"192.168.1.110",-7)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	CI	0 – 63	N/A
	slaveAddress	"0.0.0.0" – "255.255.255.255"	N/A
	logSyncInterval	-7 128pps -6 64pps -5 32pps -4 16pps -3 8pps -2 4pps -1 2pps 0 1pps 1 1/2pps 2 1/4pps 3 1/8pps 4 1/16pps	-5

Note *The first Slave added to the Master should be assigned channel index (CI) 0, for example, SetIpcPtpAdminManSlaveAddCi(0,0,"192.168.1.121",-5) assigns a slave with IP address 192.168.1.121 to channel 0 at 32 sync packets/sec.*

Note *The API should be invoked only after Ethernet link is available.*

5.4.2.20 SetIpcPtpAdminManSlaveRelCi

Description	Manually release a Slave, which is assigned to a Channel Index (CI). The Master stop sending Sync packets to the Slave and stop responding to Delay Request packets from that Slave.

```
STATUS SetIpcPtpAdminManSlaveRelCi(
    IN int32 fnId,
    IN Uint32 CI);
```

Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode		
Example	rv=SetIpcPtpAdminManSlaveRelCi(0,6)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	CI	0 – 63	N/A

Note *Upon execution of SetIpcPtpAdminManSlaveRelCi master sends CANCEL_UNICAST_TRANSMISSION(Sync,DResp) messages to the slave.*

5.4.2.21 SetIpcPtpAdminManSlaveRelAll

Description	Manually release all channels in the Master. The Master stop sending Sync packets to all the Slaves and stop responding to Delay Request packets from all the Slaves.

```
STATUS SetIpcPtpAdminManSlaveRelAll(
    IN int32 fnId);
```

Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode		
Example	rv=SetIpcPtpAdminManSlaveRelAll(0)		
Parameters	Parameter Name	Range	Default
	fnId	0	0

Note *Upon execution of SetIpcPtpAdminManSlaveRelAll master sends CANCEL_UNICAST_TRANSMISSION(Sync/DResp) message to all slaves.*

5.4.2.22 SetIpcAdminAUPRMode

Description	Set the aggregated unified packet rate mode.

```
STATUS SetIpcAdminAUPRMode(
    IN int32 fnId,
    IN int32 group,
    IN int32 AUPRMode);
```

Applicability	IPC9xxx – BC / Master, IPC17xx – Master mode			
Example	rv=SetIpcAdminAUPRMode(0,1,0)			
Parameters	Parameter Name		Range	Default
	fnId	0		0
	group	1	Group-1: Sync, FU, Delay Request and Response	N/A
		2	Group-2: Announce	
	AUPRMode	0	Unlimited	Group-1: 2
		1	Adaptive (applicable for Group-1)	Group-2: 2,0
		2	Limited	

In order to change the device settings, it is recommended to perform the following sequence after invoking the API:
SetIpcAdminStore
SetIpcPtpAdminStop
SetIpcPtpAdminStart
Or
SetIpcAdminStore
SetIpcPtpAdminReset

AUPRMode Default:

startMode	*portMapMode*	*group=1*	*group=2*
0	*0*	*2*	*2*
2	*0*	*2*	*0*
4	*0*	*2*	*2*
7	*0*	*2*	*0*
22	*0*	*2*	*0*
24	*0*	*2*	*0*
0	*0<*	*2*	*2*
2	*0<*	*2*	*0*
4	*0<*	*2*	*2*
7	*0<*	*2*	*0*
22	*0<*	*2*	*2*
24	*0<*	*2*	*2*

5.4.2.23 GetIpcAdminAUPRMode

Description	Get the aggregated unified packet rate mode.
	`STATUS GetIpcAdminAUPRMode(`
	`IN int32 fnId,`
	`IN int32 group,`
	`OUT int32* AUPRMode);`			

Applicability	IPC9xxx – BC / Master, IPC17xx – Master mode			
Example	rv=GetIpcAdminAUPRMode(0,1,&AUPRMode)			
Parameters	Parameter Name		Range	Default
	fnId	0		0
	group	See SetIpcAdminAUPRMode		N/A
	AUPRMode	See SetIpcAdminAUPRMode		

5.4.3 Clock Outputs

5.4.3.1 SetIpcAdminClkOutEn

Description	Enable or disable clock output. All clock outputs are enabled by default.
	`STATUS SetIpcAdminClkOutEn(`
	`IN int32 fnId,`
	`IN int32 clkOutIfcNmbr,`
	`IN Uint32 clkOutEn);`			

Applicability	IPC9xxx, IPC17xx, IPC1603			
Example	Enabling clock output number 1: rv=SetIpcAdminClkOutEn(0,1,1)			
Parameters	Parameter Name		Range	Default
	fnId	0		0

clkOutIfcNmbr	0	All clock outputs	0
	1	Clock Output #1	
	4	PPS Clock Output	
clkOutEn	0	Disable (three state)	1
	1	Enable	

Does not stored in ConfigDb.

5.4.3.2 GetIpcAdminClkOutEn

Description	Enable or disable clock output. All clock outputs are enabled by default.

```
STATUS GetIpcAdminClkOutEn(
    IN int32 fnId,
    OUT ClkOutEn* clkOutEn);
```

Applicability	IPC9xxx, IPC17xx, IPC1603			
Example	rv=GetIpcAdminClkOutEn(0,&clkOutEn)			
Parameters	Parameter Name		Range	Default
	fnId	0		0
	clkOutEn	See	SetIpcAdminClkOutEn	1

5.4.3.3 SetIpcAdminClkOutFreq

Description	Set a clock output frequency

```
STATUS SetIpcAdminClkOutFreq(
    IN int32 fnId,
    IN int32 clkOutFreq);
```

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603			
Example	Setting clock outputs to 2.048 MHz: rv=SetIpcAdminClkOutFreq(0,2)			
Parameters	Parameter Name		Range	Default
	fnId	0		0
	clkOutFreq	1	1.544MHz	3
		2	2.048MHz	
		3	10MHz	
		7	ETHRXCLKO1 25/125M – IPC9600 only	

The clock out frequency is automatically set to 10MHz when the IPC9xxx is set to Master mode or when IPC17xx is set to master mode. Frequency cannot be changed.

5.4.3.4 GetIpcAdminClkOutFreq

Description	Get a clock output frequency

```
STATUS GetIpcAdminClkOutFreq(
    IN int32 fnId,
    OUT int32* clkOutFreq);
```

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603			
Example	Getting clock outputs frequency: rv=GetIpcAdminClkOutFreq(0,&clkOutFreq)			
Parameters	Parameter Name		Range	Default
	fnId	0		0
	clkOutFreq	See	SetIpcAdminClkOutFreq	3

5.4.3.5 SetIpcAdminClkEthEn

Description	Set the Ethernet clock output enable.

```
STATUS SetIpcAdminClkEthEn(
    IN int32 fnId,
    IN int32 clkEthType,
    IN int32 clkEthEn);
```

Applicability	IPC9600			
Example	rv= SetIpcAdminClkEthEn(0,9,1)			
Parameters	Parameter Name		Range	Default
	fnId	0		0

clkEthType	0	ETHRXCLKO1	N/A
	1	ETHRXCLKO2	
	2	ETHREFCLK	
	9	All	
clkEthEn	0	Three-state	See note
	1	Enable	

Default values:
clkEthType=0 – Default: clkEthEn=0
clkEthType=1 – Default: clkEthEn=0
clkEthType=2 – Default: clkEthEn=1

5.4.3.6 GetIpcAdminClkEthEn

Description: Get the Ethernet clock output enable.

```
STATUS GetIpcAdminClkEthEn(
    IN int32 fnId,
    IN int32 clkEthType,
    OUT int32* clkOutFreq);
```

Applicability: IPC9600

Example: rv=GetIpcAdminClkEthEn(0,1,&clkEthEn)

Parameters:

Parameter Name		Range	Default
fnId	0		0
clkEthType	0	ETHRXCLKO1	N/A
	1	ETHRXCLKO2	
	2	ETHREFCLK	
clkEthEn	0	Three-state	See note
	1	Enable	

Default values:
clkEthType=0 – Default: clkEthEn=0
clkEthType=1 – Default: clkEthEn=0
clkEthType=2 – Default: clkEthEn=0

5.4.3.7 SetIpcAdminFreqOfs

Description: Set the Slave's frequency offset from Master's frequency (if known) in ppb.

```
STATUS SetIpcAdminFreqOfs(
    IN int32 fnId,
    IN int32 freqOfstPpb);
```

Applicability: IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603

Example: rv=SetIpcAdminFreqOfs(0,700)

Parameters:

Parameter Name	Range	Default
fnId	0	0
freqOfstPpb	-10000 – +10000	0

It is recommended to use SetIpcAdminFreqOfs only in case the slave's frequency offset from the master is known. Setting of non-accurate slaveFreqOfstPpb may increase the device acquisition time.

SetIpcAdminFreqOfs shall be invoked right after invoking SetIpcPtpAdminStart.

freqOfstPpb is calculated as:

$$freqOfstPpb = \frac{f_{OSC_{FR}} - f_{MASTER}}{f_{OSC_{FR}}} \times 10^9 [ppb]$$

Where:

$f_{OSC_{FR}}$ *is the Slave's oscillator free run frequency*

f_{MASTER} *is the Master Frequency*

5.4.3.8 GetIpcAdminFreqOfs

Description: Get the Slave's frequency offset from Master's frequency in ppb.

```
STATUS GetIpcAdminFreqOfs(
```

```
    IN int32 fnId,
    OUT int32* freqOfstPpb);
```

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=GetIpcAdminFreqOfs(0,&freqOfstPpb)

Parameters

Parameter Name	Range	Default
fnId	0	0
freqOfstPpb	See SetIpcAdminFreqOfs	0

5.4.3.9 SetIpcAdminFreqOfsMode

Description: Set the freqOfstMode. The freqOfstMode enables to use the slave's frequency offset (freqOfstPpb), as was stored in the configuration database (configDB) on the local FLASH memory, and to use it during the next power cycle or reset. The user shall invoke SetIpcAdminStore(0,0) in order to perform the store to FLASH memory.

```
STATUS SetIpcAdminFreqOfsMode (
    IN int32 fnId,
    IN int32 freqOfstMode);
```

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=SetIpcAdminFreqOfsMode(0,1)

Parameters

Parameter Name	Range		Default
fnId	0		0
freqOfstMode	0	Do not use the stored frequency	0
	1	Use the stored frequency	

5.4.3.10 GetIpcAdminFreqOfsMode

Description: Get the freqOfstMode.

```
STATUS GetIpcAdminFreqOfsMode(
    IN int32 fnId,
    OUT int32* freqOfstMode);
```

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=GetIpcAdminFreqOfsMode(0,&freqOfstMode)

Parameters

Parameter Name	Range	Default
fnId	0	0
freqOfstMode	See SetIpcAdminFreqOfsMode	0

5.4.4 Clock Inputs

Note *The pull-in range of the master is complying with stratum 3 pull-in range as specify in GR1244.*

5.4.4.1 SetIpcAdminClkInRefMode

Description: In IPC9xxx Master mode and IPC17xx Master mode: Setting the CLK IN reference and controlling clock status (clockclass) report by master:

Auto	CLK IN PLL is enabled. Clock status (clock class) is derived from CLK IN PLL only.
System	CLK IN PLL is disabled. OSCIN is used as reference clock. Clock status (clock class) is derived from the external clock source status as controlled by SetIpcAdminClkInExtSrcStat API. ClockClass is controlled by the external clock source status.
Extended	CLK IN PLL is enabled. Clock status (clock class) is a combination of both CLK IN PLL and the external clock source status as controlled by SetIpcAdminClkInExtSrcStat API.
Free run	CLK IN PLL is disabled. Master shall run in free run mode. The clockClass shall not be influenced by the SetIpcAdminClkInExtSrcStat.
Manual	CLK IN PLL is set according to SetIpcAdminClkInPllEn API. The clock class is set manually using the SetIpcPtpDsClockClass API. The ptpFrequencyTraceableFlag is set manually using SetIpcPtpDsFrequencyTraceable. SetIpcAdminClkInExtSrcStat API

		shall not be used.
	Extended auto	This mode applicable to IPC9600 only, extDev=44. CLK IN PLL is enabled. Clock status (clock class) is a combination of both CLK IN PLL and the external clock source status as retrieved by GetIpcAdminExtDeviceStatus.
	Auto-A G.8275.1	Same as 0 (Auto) = G.8275.1 Table V.3 CLK IN PLL is enabled. Clock status (clock class) is derived from CLK IN PLL only. In case device state is TR as reported by GetIpcPtpState (Gstate.state), the parentDs.clockQuality.clockClass maintains the previous ClockClass (i.e. do not change ClockClass)
	Extended auto-A G.8275.1	Same as 5 (Extended auto) = G.8275.1 Table V.3 In case device state is TR as reported by GetIpcPtpState (Gstate.state), the parentDs.clockQuality.clockClass maintains the previous ClockClass (i.e. do not change ClockClass)

```
STATUS SetIpcAdminClkInRefMode(
    IN int32 fnId,
    IN int32 clkInRefMode);
```

Applicability	IPC9xxx – Master mode, IPC17xx – Master mode
Example	rv=SetIpcAdminClkInRefMode(0,2)

Parameters

Parameter Name		Range	Default
fnId	0		0
clkInRefMode	0	Auto	0 (profile-0,1)
	1	System	6 (profile-2)
	2	Extended	
	3	Free run	
	4	Manual	
	5	Extended auto (IPC9600 only)	
	6	Auto-A	
	7	Extended auto-A (IPC9600 only)	

5.4.4.2 GetIpcAdminClkInRefMode

Description	Get the clock in reference and status report mode.

```
STATUS GetIpcAdminClkInRefMode(
    IN int32 fnId,
    OUT int32* clkInRefMode);
```

Applicability	IPC9xxx – Master mode, IPC17xx – Master mode
Example	rv=GetIpcAdminClkInRefMode(0,&clkInRefMode)

Parameters

Parameter Name		Range	Default
fnId	0		0
clkInRefMode	See SetIpcAdminClkInRefMode		

5.4.4.3 SetIpcAdminClkInFreq

Description	Select the reference clock input frequency

```
STATUS SetIpcAdminClkInFreq(
    IN int32 fnId,
    IN int32 clkInFreq);
```

Applicability	IPC9xxx – Master mode, IPC17xx – Master mode
Example	rv=SetIpcAdminClkInFreq(0,1)

Parameters

Parameter Name		Range	Default
fnId	0		0
clkInFreq	0	1PPS/1Hz	0
	1	1.544MHz	
	2	2.048MHz	
	3	10MHz	

5.4.4.4 GetIpcAdminClkInFreq

Description	Get the reference clock input frequency

```
STATUS GetIpcAdminClkInFreq(
```

```
    IN int32 fnId,
    OUT int32* clkInFreq);
```

Applicability	IPC9xxx – Master mode, IPC17xx – Master mode
Example	rv=GetIpcAdminClkInFreq(0,&clkInFreq)

Parameters

Parameter Name	Range	Default
fnId	0	0
clkInFreq	See SetIpcAdminClkInFreq	0

5.4.4.5 SetIpcAdminClkInExtSrcStat

Description	Set the external clock source status. This API should be called if the reference clock is provided by an external clock source (e.g., PLL).

```
STATUS SetIpcAdminClkInExtSrcStat(
    IN int32 fnId,
    IN int32 extClkSrcStat);
```

Applicability	IPC9xxx – Master mode, IPC17xx – Master mode
Example	rv=SetIpcAdminClkInExtSrcStat(0,1)

Parameters

Parameter Name	Range		Default
fnId	0		0
extClkSrcStat	0	Unknown	0
	1	Lock	
	4	Free run	
	8	Holdover	

5.4.4.6 GetIpcAdminClkInExtSrcStat

Description	Get the external clock source status. This API should be called if the reference clock is provided by an external clock source (e.g., GPS).

```
STATUS GetIpcAdminClkInExtSrcStat(
    IN int32 fnId,
    OUT int32* extClkSrcStat);
```

Applicability	IPC9xxx – Master mode, IPC17xx – Master mode
Example	rv=GetIpcAdminClkInExtSrcStat(0,&extClkSrcStat)

Parameters

Parameter Name	Range	Default
fnId	0	0
extClkSrcStat	See SetIpcAdminClkInExtSrcStat	0

5.4.4.7 SetIpcAdminClkInPllEn

Description	Enables or disables the clock input PLL. In case clkInPllEn =0 then the reference clock is the OSCIN signal or SYSIN.

```
STATUS SetIpcAdminClkInPllEn(
    IN int32 fnId,
    IN int32 clkInPllEn);
```

Applicability	IPC9xxx – Master mode, IPC17xx – Master mode
Example	rv=SetIpcAdminClkInPllEn(0,0)

Parameters

Parameter Name	Range		Default
fnId	0		0
clkInPllEn	0	Disable	1
	1	Enable	

5.4.4.8 GetIpcAdminClkInPllEn

Description	Get the clkInPllEn.

```
STATUS GetIpcAdminClkInPllEn(
    IN int32 fnId,
    OUT int32* clkInPllEn);
```

Applicability	IPC9xxx – Master mode, IPC17xx – Master mode
Example	rv=GetIpcAdminClkInPllEn(0,&clkInPllEn)

Parameters

Parameter Name	Range	Default
fnId	0	0
clkInPllEn	See SetIpcAdminClkInPllEn	1

5.4.4.9 SetIpcAdminClkInPllRst

Description	Reset the clock input PLL
	`STATUS SetIpcAdminClkInPllRst(`
	`  IN int32 fnId);`		

Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode		
Example	rv=SetIpcAdminClkInPllRst(0)		
Parameters	Parameter Name	Range	Default
	fnId	0	0

5.4.4.10 SetIpcAdminClkInPllRlkTh

Description	Set the maximal time (in minutes) by which the clock input PLL attempts relocking and the maximal time the master stay in holdover status. Following by clkInPllRlkTh time, the master shall move to free running state.
	`STATUS SetIpcAdminClkInPllRlkTh(`
	`  IN int32 fnId,`
	`  IN Uint32 clkInPllRlkTh);`		

Applicability	IPC9xxx – Master mode, IPC17xx – Master mode		
Example	rv=SetIpcAdminClkInPllRlkTh(0,100)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	clkInPllRlkTh	5 – 180	120

5.4.4.11 GetIpcAdminClkInPllRlkTh

Description	Get the maximal time (in minutes) by which the clock input PLL attempts relocking and the maximal time the master stay in holdover status.
	`STATUS GetIpcAdminClkInPllRlkTh(`
	`  IN int32 fnId,`
	`  OUT Uint32* clkInPllRlkTh);`		

Applicability	IPC9xxx – Master mode, IPC17xx – Master mode		
Example	rv=GetIpcAdminClkInPllRlkTh(0,&clkInPllRlkTh)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	clkInPllRlkTh	See SetIpcAdminClkInPllRlkTh	120

5.4.5 Time of Day UART

5.4.5.1 SetIpcAdminTodUart

Description	Control the ToD UART operation.
	uartEn=0 - Disable: ToD UART doesn't send or receive NMEA messages.
	uartEn=1 - Enable: ToD UART send and receive NMEA messages.
	uartEn=2 - Enable & report: ToD UART send and receive NMEA messages & enable report msg 1658.
	`STATUS SetIpcAdminTodUart(`
	`  IN int32 fnId,`
	`  IN int32 uartEn);`			

Applicability	IPC9xxx, IPC17xx, IPC1603			
Example	rv=SetIpcAdminTodUart(0,1)			
Parameters	Parameter Name		Range	Default
	fnId	0		0
	uartEn	0	Disable	1
		1	Enable	
		2	Enable & msg 1658	

The ToD UART transmit NMEA messages when in Slave mode only if Slave time is above GPS epoch, i.e., time in parent master is above GPS epoch.

5.4.5.2 GetIpcAdminTodUart

Description	Get the ToD UART operation mode.

	STATUS GetIpcAdminTodUart(**IN int32 fnId,** **OUT int32* uartEn);**
Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=GetIpcAdminTodUart(0,&uartEn)

Parameters

Parameter Name	Range	Default
fnId	0	0
uartEn	See SetIpcAdminTodUart	1

5.4.6 Network Interface

It's recommended, following using network interface configuration APIs, to perform:
SetIpcAdminStore
SetIpcPtpAdminStop
SetIpcPtpAdminStart
Or
SetIpcAdminStore
SetIpcPtpAdminReset

Only network interface 0 is supported.

5.4.6.1 SetIpcAdminIfcNetMdioData

Description	Write data to a specific register in the PHY.
	STATUS SetIpcAdminIfcNetMdioData(**IN int32 fnId,** **IN Uint16 ifcNetPhyAddr,** **IN Uint16 mdioReg,** **IN Uint16 mdioData);**
Applicability	IPC9xxx, IPC17xx, IPC1603
Example	Writing the data 0xabcd to MDIO register 0x2 in PHY #1 rv=SetIpcAdminIfcNetMdioData(0,8,0x2,0xabcd)

Parameters

Parameter Name	Range	Default
fnId	0	0
ifcNetPhyAddr	0 – 31	N/A
mdioReg	0x0 – 0x1F	N/A
mdioData	N/A	N/A

ifcNetPhyAddr 8 is reserved for the IPC9xxx/IPC17xx/IPC1603.

5.4.6.2 GetIpcAdminIfcNetMdioData

Description	Read data from a specific register in a PHY.
	STATUS GetIpcAdminIfcNetMdioData(**IN int32 fnId,** **IN Uint16 ifcNetPhyAddr,** **IN Uint16 mdioReg,** **OUT Uint16* mdioData);**
Applicability	IPC9xxx, IPC17xx, IPC1603
Example	Reading MDIO register 0x2 from PHY #1: rv=GetIpcAdminIfcNetMdioData(0,1,0x2,&mdioData)

Parameters

Parameter Name	Range	Default
fnId	0	0
ifcNetPhyAddr	See SetIpcAdminIfcNetMdioData	N/A
mdioReg	See SetIpcAdminIfcNetMdioData	N/A
mdioData	See SetIpcAdminIfcNetMdioData	N/A

ifcNetPhyAddr 8 is reserved for the IPC9xxx/IPC17xx/IPC1603

5.4.6.3 SetIpcAdminIfcNetMacAddr

Description	Set the network interface MAC address.
	`STATUS SetIpcAdminIfcNetMacAddr(`
	`IN int32 fnId,`
	`IN int32 ifcNetNmbr,`
	`IN char* ifcNetMacAddr);`

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=SetIpcAdminIfcNetMacAddr(0,0,"1A:2B:3C:4D:5E:6F")

Parameters

Parameter Name	Range		Default
fnId	0		0
ifcNetNmbr	0		0
ifcNetMacAddr	N/A	Slave	00:0A:35:01:F4:14
		Master	00:0A:35:01:F4:15
		BC	00:0A:35:01:F4:16

It is recommended to perform SetIpcAdminIfcNetIpAddr (even if IP address was not changed) right after setting new MAC address using SetIpcAdminIfcNetMacAddr. It will generate Gratuitous ARP.

5.4.6.4 GetIpcAdminIfcNetMacAddr

Description	Get the network interface MAC address
	`STATUS GetIpcAdminIfcNetMacAddr(`
	`IN int32 fnId,`
	`IN int32 ifcNetNmbr,`
	`OUT char** ifcNetMacAddr);`

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=GetIpcAdminIfcNetMacAddr(0,0,&ifcNetMacAddr)

Parameters

Parameter Name	Range	Default
fnId	0	0
ifcNetNmbr	See SetIpcAdminIfcNetMacAddr	0
ifcNetMacAddr	See SetIpcAdminIfcNetMacAddr	

5.4.6.5 SetIpcAdminIfcNetMacSpeed

Description	Set the network interface MAC speed manually.
	`STATUS SetIpcAdminIfcNetMacSpeed(`
	`IN int32 fnId,`
	`IN Uint16 ifcNetNmbr,`
	`IN Uint16 ifcMacSpeed);`

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=SetIpcAdminIfcNetMacSpeed(0,0,1)

Parameters

Parameter Name	Range		Default
fnId	0		0
ifcNetNmbr	0		0
ifcMacSpeed	1	100Mbps	2
	2	1000Mbps	

5.4.6.6 GetIpcAdminIfcNetMacSpeed

Description	Get the network interface speed.
	`STATUS GetIpcAdminIfcNetMacSpeed(`
	`IN int32 fnId,`
	`IN Uint16 ifcNetNmbr,`
	`OUT Uint16* ifcMacSpeed);`

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=GetIpcAdminIfcNetMacSpeed(0,0,&ifcMacSpeed)

Parameters

Parameter Name	Range	Default
fnId	0	0
ifcNetNmbr	0	0
ifcMacSpeed	See SetIpcAdminIfcNetMacSpeed	2

5.4.6.7 SetIpcAdminIfcNetMacEn

Description	Enable or disable network interface.
	`STATUS SetIpcAdminIfcNetMacEn(`
	`  IN int32 fnId,`
	`  IN int32 ifcNetNmbr,`
	`  IN int32 ifcNetEn);`			

Applicability	IPC9xxx, IPC17xx, IPC1603			
Example	rv=SetIpcAdminIfcNetMacEn(0,0,0)			
Parameters	Parameter Name		Range	Default
	fnId	0		0
	ifcNetNmbr	0		0
	ifcNetEn	0	Disable	1
		1	Enable	
		2	Enable Tx only	
		3	Enable Rx only	

5.4.6.8 GetIpcAdminIfcNetMacEn

Description	Get ifcNetEn.
	`STATUS GetIpcAdminIfcNetMacEn(`
	`  IN int32 fnId,`
	`  IN int32 ifcNetNmbr,`
	`  OUT int32* ifcNetEn);`			

Applicability	IPC9xxx, IPC17xx, IPC1603			
Example	rv=GetIpcAdminIfcNetMacEn(0,0,&ifcNetEn)			
Parameters	Parameter Name		Range	Default
	fnId	0		0
	ifcNetNmbr	0		0
	ifcNetEn	See SetIpcAdminIfcNetMacEn		

5.4.6.9 SetIpcAdminIfcNetPhyClkMode

Description	Set the Ethernet PHY clock mode in GE. This is a PHY dependent API used for IPC500 only.
	`STATUS SetIpcAdminIfcNetPhyClkMode(`
	`  IN int32 fnId,`
	`  IN int32 ethPhyClkMode);`			

Applicability	IPC9xxx			
Example	rv=SetIpcAdminIfcNetPhyClkMode(0,1)			
Parameters	Parameter Name		Range	Default
	fnId	0		0
	ethPhyClkMode	0	No Action	0
		1	Auto	
		2	Master	
		3	Slave	

Note *GE – It is under the user responsibility to ensure both sides should not configure as Slave-Slave and not as Master-Master. In such a case there is no Link.*

Note *FE – Setting does not works, one side is always Master and the other side is always Auto.*

5.4.6.10 GetIpcAdminIfcNetPhyClkMode

Description	Get the Ethernet PHY clock mode in GE. This is a PHY dependent API used for IPC500 only.
	`STATUS GetIpcAdminIfcNetPhyClkMode(`
	`  IN int32 fnId,`
	`  IN int32 ethPhyClkRdHw,`
	`  OUT int32* ethPhyClkMode);`

Applicability	IPC9xxx
Example	rv=GetIpcAdminIfcNetPhyClkMode(0,0,&ethPhyClkMode)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	ethPhyClkRdHw	0 – Read from PHY in case mirror device register ethPhyClkMode=1,2,3	0
		1 – Read from PHY regardless mirror device register ethPhyClkMode value	
	ethPhyClkMode	See SetIpcAdminIfcNetPhyClkMode	0

5.4.6.11 GetIpcAdminIfcNetPhyClkStatus

Description	Get the Ethernet PHY clock status. This is a PHY dependent API used for IPC300/IPC400/IPC400A/IPC500 only.

```
STATUS GetIpcAdminIfcNetPhyClkStatus(
    IN int32 fnId,
    OUT int32* ethPhyClkStatus);
```

Applicability	IPC9xxx
Example	rv=GetIpcAdminIfcNetPhyClkStatus(0,&ethPhyClkStatus)

Parameters	Parameter Name	Range		Default
	fnId	0		0
	ethPhyClkStatus	0	Master	N/A
		1	Slave	

5.4.6.12 SetIpcAdminIfcNetPhyReset

Description	Resets the Ethernet PHY device via the PERRSTn pin

```
STATUS SetIpcAdminIfcNetPhyReset(
    IN int32 fnId
    IN int32 netEthPhyMode);
```

Applicability	IPC9xxx
Example	rv=SetIpcAdminIfcNetPhyReset(0,0)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	netEthPhyMode	0 – Reset	0
		1 – Set Reset pin to high (1)	
		2 – Set Reset pin to low (0)	

5.4.6.13 SetIpcAdminIfcNetIpAddr

Description	Set network interface IP address.

```
STATUS SetIpcAdminIfcNetIpAddr(
    IN int32 fnId,
    IN int32 ifcNetNmbr,
    IN char* ifcNetIpAddr);
```

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=SetIpcAdminIfcNetIpAddr(0,0,”192.168.10.1”) rv=SetIpcAdminIfcNetIpAddr(0,0,"0xC0.0xA8.0x0A.0x01") rv=SetIpcAdminIfcNetIpAddr(0,0,"0300.0250.012.01")

Parameters	Parameter Name	Range		Default	
	fnId	0		0	
	ifcNetNmbr	0		0	
	ifcNetIpAddr	N/A		Slave	192.168.1.20
				Master	192.168.1.21
				BC	192.168.1.22

Note *SetIpcAdminIfcNetIpAddr is been followed by Gratuitous ARP.*

Note *It is recommended to perform SetIpcAdminIfcNetIpAddr (even if IP address was not changed) right after setting new MAC address using SetIpcAdminIfcNetMacAddr. It will generate Gratuitous ARP.*

Note *The SetIpcAdminIfcNetIpAddr converts the ifcNetIpAddr Internet host address of numbers-and-dots notation into binary data in network byte order. The components of the dotted address can be specified in decimal, octal (with a leading 0), or hexadecimal, (with a leading 0x). This conversion is similar to the conversion performed by the known inet_addr( ) function.*

The ifcNetIpAddr shall be properly selected. The ifcNetIpAddr shall be unicast address only.

5.4.6.14 GetIpcAdminIfcNetIpAddr

Description	Get network interface IP address.

```
STATUS GetIpcAdminIfcNetIpAddr(
    IN int32 fnId,
    IN int32 ifcNetNmbr,
    OUT char** ifcNetIpAddr);
```

Applicability	IPC9xxx, IPC17xx, IPC1603		
Example	rv=GetIpcAdminIfcNetIpAddr(0,0,ifcNetIpAddr)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	ifcNetNmbr	0	0
	ifcNetIpAddr	See SetIpcAdminIfcNetIpAddr	

5.4.6.15 SetIpcAdminIfcNetIpSubnetMask

Description	Set the network interface's subnet mask.

```
STATUS SetIpcAdminIfcNetIpSubnetMask(
    IN int32 fnId,
    IN int32 ifcNetNmbr,
    IN char* ifcNetIpSubnetMask);
```

Applicability	IPC9xxx, IPC17xx, IPC1603		
Example	rv=SetIpcAdminIfcNetIpSubnetMask(0,0,"255.255.255.0")		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	ifcNetNmbr	0	0
	ifcNetIpSubnetMask	N/A	255.255.255.0

5.4.6.16 GetIpcAdminIfcNetIpSubnetMask

Description	Get the network interface's subnet mask.

```
STATUS GetIpcAdminIfcNetIpSubnetMask(
    IN int32 fnId,
    IN int32 ifcNetNmbr,
    OUT char** ifcNetIpSubnetMask);
```

Applicability	IPC9xxx, IPC17xx, IPC1603		
Example	rv=GetIpcAdminIfcNetIpSubnetMask(0,0,&ifcNetIpSubnetMask)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	ifcNetNmbr	0	0
	ifcNetIpSubnetMask	N/A	255.255.255.0

5.4.6.17 SetIpcAdminIfcNetIpDefaultGateway

Description	Set default gateway IP address.

```
STATUS SetIpcAdminIfcNetIpDefaultGateway(
    IN int32 fnId,
    IN int32 ifcNetNmbr,
    IN char* ifcNetIpDefaultGateway);
```

Applicability	IPC9xxx, IPC17xx, IPC1603		
Example	rv=SetIpcAdminIfcNetIpDefaultGateway(0,0,"192.168.1.1")		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	ifcNetNmbr	0	0
	ifcNetIpDefaultGateway	N/A	192.168.1.1

5.4.6.18 GetIpcAdminIfcNetIpDefaultGateway

Description	Get default gateway IP address.

```
STATUS GetIpcAdminIfcNetIpDefaultGateway(
    IN int32 fnId,
    IN int32 ifcNetNmbr,
    OUT char** ifcNetIpDefaultGateway);
```

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=GetIpcAdminIfcNetIpDefaultGateway(0,0,&ifcNetIpDefaultGateway)

Parameters

Parameter Name	Range	Default
fnId	0	0
ifcNetNmbr	0	0
ifcNetIpDefaultGateway	N/A	192.168.1.1

5.4.6.19 GetIpcAdminIfcNetConfig

Description: Get network interface configuration.

```
STATUS GetIpcAdminIfcNetConfig(
    IN int32 fnId,
    IN int32 ifcNetNmbr,
    OUT IfcNetConf** ifcNetConf);
```

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=GetIpcAdminIfcNetConfig(0,0,&ifcNetConf)

Parameters

Parameter Name	Range	Default
fnId	0	0
ifcNetNmbr	0	0
ifcNetConf	N/A	N/A

5.4.6.20 SetIpcAdminIfcNetArpTableClear

Description: Clear the ARP table.

```
STATUS SetIpcAdminIfcNetArpTableClear(
    IN int32 fnId
    IN int32 mode);
```

Applicability	IPC9xxx, IPC17xx
Example	rv=SetIpcAdminIfcNetArpTableClear(0,0)

Parameters

Parameter Name	Range	Default
fnId	0	0
mode	0 – Clear automatic entries	
	1 – Clear automatic & static entries	

5.4.6.21 SetIpcAdminIfcNetArpAdd

Description: Add or update a static entry in ARP table. Static entries does not aged.

```
STATUS SetIpcAdminIfcNetArpAdd(
    IN int32 fnId,
    IN char* ipAddr,
    IN char* ifcNetMacAddr);
```

Applicability	IPC9xxx, IPC17xx
Example	rv=SetIpcAdminIfcNetArpAdd(0,"192.168.1.21","00:0A:35:01:F4:15")

Parameters

Parameter Name	Range	Default
fnId	0	0
ipAddr	IP address range	N/A
ifcNetMacAddr	MAC address range	N/A

5.4.6.22 SetIpcAdminIfcNetArpDel

Description: Delete a static entry from the ARP table.

```
STATUS SetIpcAdminIfcNetArpDel(
    IN int32 fnId,
    IN char* ipAddr);
```

Applicability	IPC9xxx, IPC17xx
Example	rv=SetIpcAdminIfcNetArpDel(0,"192.168.1.21")

Parameters

Parameter Name	Range	Default
fnId	0	0

	ipAddr	IP address range	N/A

5.4.6.23 GetIpcAdminIfcNetArpTable

Description	Show ARP table. arpTable max size is 80 entries, each page includes 8 entries. **STATUS GetIpcAdminIfcNetArpTable(** **IN int32 fnId,** **IN int32 type,** **IN int32 page,** **OUT IpcEthArpEntry* arpTable);**
Applicability	IPC9xxx, IPC17xx
Example	rv=GetIpcAdminIfcNetArpTable(0,0,0,&arpTable)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	type	0 - automatic entries	0
		1 - static entries	
		2 - all entries	
	page	0 - 9	0
	arpTable	N/A	N/A

IpcEthArpEntry.state
0 – Empty
1 – Pending
2 – Stable
IpcEthArpEntry.type
0 – Automatic
1 – Static

Examples:

GetIpcAdminIfcNetArpTable(0,0,0)

```
================================================
                  Arp Table
================================================
IP Address      Mac Address          State    Type
------------------------------------------------
192.168.1.21    00:0A:35:01:F4:15    2        0
------------------------------------------------
```

GetIpcAdminIfcNetArpTable(0,1,0)

```
================================================
                  Arp Table
================================================
IP Address      Mac Address          State    Type
------------------------------------------------
192.168.1.31    00:0A:35:01:F4:0F    2        1
192.168.1.23    00:0A:35:01:F4:17    2        1
192.168.1.24    00:0A:35:01:F4:18    2        0
192.168.1.25    00:0A:35:01:F4:19    2        1
192.168.1.26    00:0A:35:01:F4:0A    2        0
192.168.1.27    00:0A:35:01:F4:0B    2        1
192.168.1.28    00:0A:35:01:F4:0C    2        1
192.168.1.29    00:0A:35:01:F4:0D    2        1
------------------------------------------------
```

GetIpcAdminIfcNetArpTable(0,1,1)

```
================================================
                  Arp Table
================================================
IP Address      Mac Address          State    Type
------------------------------------------------
192.168.1.30    00:0A:35:01:F4:0E    2        1
------------------------------------------------
```

Aging time:
IpcEthArpEntry.type = 0 (Automatic): 5 minutes
IpcEthArpEntry.type = 1 (Static): Never

5.4.6.24 GetIpcAdminIfcNetArpShow

Description	Show ARP entry. This API returns the MAC address. In case IP address is not in ARP table the API return -7. (empty)

```
STATUS GetIpcAdminIfcNetArpShow(
    IN int32 fnId,
    IN char* ipAddr,
    OUT char* ifcNetMacAddr);
```

Applicability	IPC9xxx, IPC17xx
Example	rv=GetIpcAdminIfcNetArpShow(0,"192.168.1.21",&ifcNetMacAddr)

Parameters

Parameter Name	Range	Default
fnId	0	0
ipAddr	IP address range	N/A
ifcNetMacAddr	MAC address range	N/A

Examples:

GetIpcAdminIfcNetArpShow(0,"192.168.1.24")
00:00:02:48 [#1100] [GetIpcAdminArpShow]: 00:0A:35:01:F4:18
result = OK

GetIpcAdminIfcNetArpShow(0,"192.168.1.22")
00:00:16:32 [#1100] [GetIpcAdminIfcNetArpShow]: 00:00:00:00:00:00
result = EMPTY

5.4.6.25 SetIpcAdminIfcNetArpServedMode

Description	Set the arpServedMode. arpServedMode=0 – The master & slave ports perform normal ARP operation, in which automatic ARP entries have aging time of 5 minutes. arpServedMode=1 – The master & slave ports shall keep ARP entries based on PTP operation. Master port: The master shall not resent additional ARP after the first ARP, as long as the PTP channel is active. The SlavesServed, presents the channels, and is available by using GetIpcPtpAdminSlavesServed. The process is as follow: Every 2 minutes the device clears the ARP expiration counter of all entries available at SlavesServed. In case of channel aging from SlaveServed table, the ARP expiration counter will not be cleared (i.e. last clear was performed 0-2 minutes before the aging). As a result the aging from the ARP table, which takes 5 minutes, shall be performed within 3-5 minutes from aging from SlaveServed table. Slave port: The slave shall not resent additional ARP to the selected master after the first ARP, as long as the selected master will continue to be available. The selected master reported by GetIpcPtpBmcMasterSelected. The process is as follow: Every 2 minutes the device clears the ARP expiration counter of the entry identical to the selected master. In case of master is no longer selected, the ARP expiration counter will not be cleared (i.e. last clear was performed 0-2 minutes before). As a result the aging from the ARP table, which takes 5 minutes, shall be performed within 3-5 minutes from master aging.

```
STATUS SetIpcAdminIfcNetArpMode(
    IN int32 fnId,
    IN int32 arpServedMode);
```

Applicability	IPC9xxx, IPC17xx
Example	rv=SetIpcAdminIfcNetArpServedMode(0,1)

Parameters

Parameter Name		Range	Default
fnId	0		0
arpServedMode	0	Disable	1

	1	Enable – Master & Slave ports

5.4.6.26 GetIpcAdminIfcNetArpServedMode

Description	Get the arpServedMode.
	`STATUS GetIpcAdminIfcNetArpServedMode(`
	`IN int32 fnId,`
	`OUT int32* arpServedMode);`

Applicability	IPC9xxx, IPC17xx
Example	rv=GetIpcAdminIfcNetArpServedMode(0,&arpServedMode)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	arpServedMode	See SetIpcAdminIfcNetArpServedMode	

5.4.6.27 SetIpcAdminIfcNetArpDGMode

Description	Set the arpDGMode. While enabled, the device send ARP packet to DefaultGateway IP address every 1 minute.
	`STATUS SetIpcAdminIfcNetArpDGMode(`
	`IN int32 fnId,`
	`IN int32 arpDGMode);`

Applicability	IPC9xxx, IPC17xx
Example	rv=SetIpcAdminIfcNetArpDGMode(0,1)

Parameters	Parameter Name	Range		Default
	fnId	0		0
	arpDGMode	0	Disable	0
		1	Enable	

Note *Regardless arpDGMode the device send ARP packet to DefaultGateway IP address in accordance to standard ARP operation.*

5.4.6.28 GetIpcAdminIfcNetArpDGMode

Description	Get the arpDGMode.
	`STATUS GetIpcAdminIfcNetArpDGMode(`
	`IN int32 fnId,`
	`OUT int32* arpDGMode);`

Applicability	IPC9xxx, IPC17xx
Example	rv=GetIpcAdminIfcNetArpDGMode(0,&arpDGMode)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	arpDGMode	See SetIpcAdminIfcNetArpDGMode	

5.4.6.29 SetIpcAdminIfcNetArpGrt

Description	Set the arpGrt.
	`STATUS SetIpcAdminIfcNetArp(`
	`IN int32 fnId,`
	`IN int32 arpGrt);`

Applicability	IPC9xxx, IPC17xx
Example	rv=SetIpcAdminIfcNetArpGrt(0,15)

Parameters	Parameter Name	Range		Default
	fnId	0		0
	arpGrt	0	Disable	0
		1-20	Send Gratuitous ARP packet every arpGrt minutes	

5.4.6.30 GetIpcAdminIfcNetArpGrt

Description	Get the arpGrt.

	STATUS GetIpcAdminIfcNetArpGrt(**IN int32 fnId,** **OUT int32* arpGrt);**
Applicability	IPC9xxx, IPC17xx
Example	rv=GetIpcAdminIfcNetArpGrt(0,&arpGrt)

Parameters

Parameter Name	Range	Default
fnId	0	0
arpGrt	See SetIpcAdminIfcNetArpGrt	

5.4.7 Network Interface Packet Report

5.4.7.1 SetIpcAdminIfcNetPerPktRepEn

Description	Enable or Disable the periodic packet report for network interface every 60 minutes. This API works only in terminal mode as been controlled by SetIpcAdminMgmtMode API. This API doesn't influence on the operation of the following APIs: GetIpcAdminIfcNetPktRep1Min/15Min/60Min/Inf. **STATUS SetIpcAdminIfcNetPerPktRepEn(** **IN int32 fnId,** **IN int32 ifcNetNmbr,** **IN int32 repEn);**
Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=SetIpcAdminIfcNetPerPktRepEn(0,0,1)

Parameters

Parameter Name	Range		Default
fnId	0		0
ifcNetNmbr	0		0
repEn	0	Disable	0
	1	Enable	

5.4.7.2 GetIpcAdminIfcNetPerPktRepEn

Description	Get the periodic packet report enable/disable status. **STATUS GetIpcAdminIfcNetPerPktRepEn(** **IN int32 fnId,** **IN int32 ifcNetNmbr,** **OUT int32* repEn);**
Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=GetIpcAdminIfcNetPerPktRepEn(0,0,&repEn)

Parameters

Parameter Name	Range	Default
fnId	0	0
ifcNetNmbr	0	0
repEn	See SetIpcAdminIfcNetPerPktRepEn	0

5.4.7.3 GetIpcAdminIfcNetPktRep1Min

Description	Get the packet report for an observation time of 1 minute. **STATUS GetIpcAdminIfcNetPktRep1Min(** **IN int32 fnId,** **IN int32 ifcNetNmbr,** **OUT PktStatis* pktStatistics);**
Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=GetIpcAdminIfcNetPktRep1Min(0,0,&pktStatistics)

Parameters

Parameter Name	Range	Default
fnId	0	0
ifcNetNmbr	0	0
pktStatistics	N/A	0

Note *The report is been updated once per minute.*

5.4.7.4 GetIpcAdminIfcNetPktRep15Min

Description	Get the packet report for an observation time of 15 minute.

```
STATUS GetIpcAdminIfcNetPktRep15Min(
    IN int32 fnId,
    IN int32 ifcNetNmbr,
    OUT PktStatis* pktStatistics);
```

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=GetIpcAdminIfcNetPktRep15Min(0,0,&pktStatistics)

Parameters

Parameter Name	Range	Default
fnId	0	0
ifcNetNmbr	0	0
pktStatistics	N/A	0

Note *The report is been updated once per minute.*

5.4.7.5 GetIpcAdminIfcNetPktRep60Min

Description: Get the packet report for an observation time of 60 minute.

```
STATUS GetIpcAdminIfcNetPktRep60Min(
    IN int32 fnId,
    IN int32 ifcNetNmbr,
    OUT PktStatis* pktStatistics);
```

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=GetIpcAdminIfcNetPktRep60Min(0,0,&pktStatistics)

Parameters

Parameter Name	Range	Default
fnId	0	0
ifcNetNmbr	0	0
pktStatistics	N/A	0

Note *The report is been updated once per minute.*

5.4.7.6 GetIpcAdminIfcNetPktRepInf

Description: Get the network report since the beginning of starting gathering statistics or since the last time the statistics was reset.

```
STATUS GetIpcAdminIfcNetPktRepInf(
    IN int32 fnId,
    IN int32 ifcNetNmbr,
    OUT PktStatis* pktStatistics);
```

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=GetIpcAdminIfcNetPktRepInf(0,0,&pktStatistics)

Parameters

Parameter Name	Range	Default
fnId	0	0
ifcNetNmbr	0	0
pktStatistics	N/A	0

Note *The report is been updated once per minute.*

5.4.7.7 SetIpcAdminIfcNetPktRepClr

Description: Reset the packet report for network interface.

```
STATUS SetIpcAdminIfcNetPktRepClr(
    IN int32 fnId,
    IN int32 ifcNetNmbr);
```

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=SetIpcAdminIfcNetPktRepClr(0,0)

Parameters

Parameter Name	Range	Default
fnId	0	0
ifcNetNmbr	0	0

5.4.7.8 SetIpcAdminIfcNetPktDumpMode

Description: Set the packet dump mode.

```
STATUS SetIpcAdminIfcNetPktDumpMode(
    IN int32 fnId,
```

```
    IN int32 RxTx,
    IN int32 MessageTypeEn,
    IN int32 MessageType,
    IN int32 LocalTSDump,
    IN int32 Number);
```

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=SetIpcAdminIfcNetPktDumpMode(0,1,1,1,1,10)

Parameters

Parameter Name	Range	Default
fnId	0	0
RxTx	0 – Rx (Ingress) 1 – Tx (Egress)	0
TypeEn	0 – Disable type filtration 1 – Enable type filtration	0
MessageType	0 – Sync 1 – Delay Req 8 – FU 9 – Delay Resp 11 – Announce 12 – Signaling 13 – Management	0
LocalTSDump	0 – Disable 1 – Enable for Rx Sync & Delay Req packets	0
Number	1 – 15 packets	1

Note

LocalTSDump enables to dump the following local Time stamps:
a. T2 for Rx Sync
b. T4 for Rx Delay Request

5.4.7.9 GetIpcAdminIfcNetPktDumpMode

Description	Get the packet dump mode.

```
STATUS GetIpcAdminIfcNetPktDumpMode(
    IN int32 fnId,
    OUT PktDumpMode* pktDumpMode);
```

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=GetIpcAdminIfcNetPktDumpMode(0,&pktDumpMode)

Parameters

Parameter Name	Range	Default
fnId	0	0
pktDumpMode	N/A	0

5.4.7.10 GetIpcAdminIfcNetPktDump

Description	Get the next packets dump. This API returns to the device log the PTP header in hexadecimal format. For Rx packets, SequenceId and local timestamp can be also reported.

```
STATUS GetIpcAdminIfcNetPktDump(
    IN int32 fnId);
```

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	**Example 1:** **SetIpcAdminIfcNetPktDumpMode(0,1,0,0,0,0,1)** **GetIpcAdminIfcNetPktDump(0)** MsgRx: 1 Type: 0x00 00: 0002 002C 0000 0400 0000 0000 0000 0000 10: 0000 0000 0000 00FF FE00 0015 0002 698F 20: 007F 0000 0000 334C 1C59 2583 **Example 2:** **SetIpcAdminIfcNetPktDumpMode(0,1,0,0,0,1,1)** **GetIpcAdminIfcNetPktDump(0)** MsgRx: 1 Type: 0x00 00: 0002 002C 0000 0400 0000 0000 0000 0000 10: 0000 0000 0000 00FF FE00 0015 0002 BDF3 20: 007F 0000 0002 BDEF 2F42 C4E7 SeqId = BDF3 LocalTS-T2 = 2BDEF:2F42FD0E

Example 3:
SetIpcAdminIfcNetPktDumpMode(0,1,0,0,0,1,5)
GetIpcAdminIfcNetPktDump(0)

MsgRx: 1 Type: 0x00
00: 0002 002C 0000 0400 0000 0000 0000 0000
10: 0000 0000 0000 00FF FE00 0015 0002 CA50
20: 007F 0000 0002 BE52 29AE 917D
SeqId = CA50
LocalTS-T2 = 2BE52:29AEC940

MsgRx: 2 Type: 0x09
00: 0902 0036 0000 0400 0000 0000 0000 0000
10: 0000 0000 0000 00FF FE00 0015 0002 D54F
20: 037F 0000 0002 BE52 29D7 2944 0000 00FF
30: FE00 0014 0001

MsgRx: 3 Type: 0x00
00: 0002 002C 0000 0400 0000 0000 0000 0000
10: 0000 0000 0000 00FF FE00 0015 0002 CA51
20: 007F 0000 0002 BE52 2B8A B7D3
SeqId = CA51
LocalTS-T2 = 2BE52:2B8AEFD2

MsgRx: 4 Type: 0x00
00: 0002 002C 0000 0400 0000 0000 0000 0000
10: 0000 0000 0000 00FF FE00 0015 0002 CA52
20: 007F 0000 0002 BE52 2D66 A585
SeqId = CA52
LocalTS-T2 = 2BE52:2D66DD48

MsgRx: 5 Type: 0x00
00: 0002 002C 0000 0400 0000 0000 0000 0000
10: 0000 0000 0000 00FF FE00 0015 0002 CA53
20: 007F 0000 0002 BE52 2F43 1203
SeqId = CA53
LocalTS-T2 = 2BE52:2F434A20

Parameters	Parameter Name	Range	Default
	fnId	0	0

This API returns the packet to the device log only.

By default, GetIpcAdminIfcNetPktDump return the next Rx packet.

SetIpcAdminIfcNetPktDumpMode clears the trigger for GetIpcAdminIfcNetPktDump.

5.4.8 Local Management Interface

5.4.8.1 SetIpcAdminMgmtMode

Description	Set the management mode.
	`STATUS SetIpcAdminMgmtMode(`
	`  IN int32 fnId,`
	`  IN Int32 mode);`

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=SetIpcAdminMgmtMode(0,1)

Parameters	Parameter Name		Range	Default
	fnId	0		0
	Mode	0	Terminal – UART	0
		1	Host – UART	
		7	Host without CRC (Rx/Tx) – UART	

In case SetIpcAdminMgmtMode(0,1) is performed, the management log information is been cleared by invoking SetIpcAdminLogClr(0).

SetIpcAdminMgmtMode(0,0) cause the device to ignore the CRC and to move to Terminal mode. It can be used in case user is in Host mode and wish to switch to Terminal mode using Terminal API.*

For robust management, API messages contain CRC16. CRC is added to end of each messages (string contains name of API with parameters) sending by the Host to target device. CRC length is 4 hexadecimal digits in ASCII. The device adds CRC16 to each response message it returns to Host besides messages contains log data. The receiving side checks the CRC. In case of the mismatch ERROR -4 (target) and ERROR -5 (Host) shall be returned.

5.4.8.2 GetIpcAdminMgmtMode

Description	Get the management mode.
	`STATUS GetIpcAdminMgmtMode(`
	`  IN int32 fnId,`
	`  OUT Int32* mode);`

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=GetIpcAdminMgmtMode(0,&mode)

Parameters

Parameter Name	Range	Default
fnId	0	0
Mode	See SetIpcAdminMgmtMode	0

5.4.8.3 SetIpcPtpAdminMsgLvl

Description	Set the message level to be displayed and written to the log. The message level is set by setting a bit (or bits) to '1' according to the required message level.
	`STATUS SetIpcPtpAdminMsgLvl(`
	`  IN int32 fnId,`
	`  IN Uint32 msgLevel);`

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=SetIpcPtpAdminMsgLvl(0,73471) - Set message level for the extended message level for BC and Master modes. rv=SetIpcPtpAdminMsgLvl(0,3615) - Set message level for periodic messages, Remote support, Administrative API (bits 0,4,5,6,7,9,10,11 are set to '1') rv=SetIpcPtpAdminMsgLvl(0,3647) - Set message level for the combination of both periodic messages, informational messages and Remote support, Administrative API (bit 0,1,4,5,6,7,9,10,11 are set to '1') rv=SetIpcPtpAdminMsgLvl(0,0), (0x0000) - Inhibits all messages.

Parameters

Parameter Name		Range		Default
fnId	0		0	
msgLevel	bit	Message Type		
	0	Periodic messages Msg [#00xx], e.g. 0001,0050	OC Slave: OC Master:	7423 (0x1CFF) 6911 (0x1AFF)
	1	Informational messages Msg [#01xx], e.g. 0101,0102	BC:	32494 (0x7EEE)
	2	Warning messages Msg [#02xx], e.g. 0201		
	3	Error messages Msg [#03xx]		
	4	Remote support Set to '1' for Slave & Master Msg [#04xx]		
	5	Remote support Set to '1'. Msg [#05xx]		
	6	Remote support Set to '1'. Msg [#06xx]		
	7	Remote support Set to '1'. Msg [#07xx]		
	8	Remote support Set to '0'. Msg [#08xx]		
	9	Remote support Set to '1' for Master & BC Msg [#09xx]		

10	Remote support Set to '1' for Slave & BC Msg [#10xx]
11	Administrative API Msg [#11xx]
12	BIST Msg [#12xx]. Msg # - not shown
13	BC Periodic messages Msg [#13xx], e.g. 1350
14	Remote support Set to '1' for BC. Msg [#14xx]
15	Remote support Set to '0'. Msg [#15xx]
16	Remote support Set to '0'. Msg [#16xx]

Note *This API is shall be used for controlling message output level to a management console (terminal).*

Note *In BC/Master, higher msgLevel is recommended only in case the Sync packet rate is 32 pps or less.*

5.4.8.4 GetIpcPtpAdminMsgLvl

Description	Get the current message level to be displayed and written to the log.
	`STATUS GetIpcPtpAdminMsgLvl(`
	`  IN int32 fnId,`
	`  OUT Uint32* msgLevel);`

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=GetIpcPtpAdminMsgLvl(0,&msgLevel)

Parameters	Parameter	Range	Default
	fnId	0	0
	msgLevel	See SetIpcPtpAdminMsgLvl	

Note *This API shall be used for getting the message output level to a management console (terminal).*

5.4.8.5 SetIpcAdminMgmtLogMode

Description	Set the management log information mode as retrieved by GetIpcAdminLog. In case LogMode=0 the log information includes all API calls. In case LogMode=1 the log information includes API Set calls. In case LogMode=2 the log information includes API Get calls. In case LogMode=3 the log information does not include the API calls.
	`STATUS SetIpcAdminMgmtLogMode(`
	`  IN int32 fnId,`
	`  IN Int32 logMode);`

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=SetIpcAdminMgmtLogMode(0,3)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	LogMode	0 The log information includes API calls 1 The log information includes API Set calls 2 The log information includes API Get calls 3 The log information doesn't include API calls	1

5.4.8.6 GetIpcAdminMgmtLogMode

Description	Get the management log information mode.
	`STATUS GetIpcAdminMgmtLogMode(`
	`  IN int32 fnId,`
	`  OUT Int32* logMode);`

Applicability	IPC9xxx, IPC17xx, IPC1603		
Example	rv=GetIpcAdminMgmtLogMode(0,&logMode)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	LogMode	See SetIpcAdminMgmtLogMode	

5.4.8.7 GetIpcAdminLog

Description	Get the log information. For proper operation, it's recommended to use this API every 10 sec.
	`STATUS GetIpcAdminLog(`
	`  OUT char* logSnippet);`

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=GetIpcAdminLog(&logSnippet)

Parameters	Parameter Name	Range	Default
	logSnippet	N/A	N/A

5.4.8.8 GetIpcAdminLogStatus

Description	Get the log status information. For proper operation, it's recommended to use this API every 60 sec.
	`STATUS GetIpcAdminLogStatus(`
	`  OUT txSysLogBufferStatus* logStatus);`

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=GetIpcAdminLogStatus(&logStatus)

Parameters	Parameter Name	Range	Default
	logStatus	N/A	N/A

The Log buffer size is 20000 chars. Under normal conditions the log utilization percentage expected to be <10%. It is recommended to use the GetIpcAdminLog API every 10sec, and to use the GetIpcAdminLogStatus API every 60sec in order to validate the log utilization percentage level. In case the log utilization percentage is higher than 10% it is recommended to increase the rate of calling to GetIpcAdminLog to one per 6sec. It's not recommended to call this API using a higher rate than every 60sec.

In case of log buffer overrun, the TxSysLogBufferStatus.logOverflow shall be set to 1 and the entire buffer shall be cleared.

5.4.8.9 SetIpcAdminLogClr

Description	Clear the log information buffer.
	`STATUS SetIpcAdminLogClr(`
	`  IN int32 fnId);`

Applicability	IPC9xxx, IPC17xx, IPC1603
Example	rv=SetIpcAdminLogClr(0)

Parameters	Parameter Name	Range	Default
	fnId	0	0

5.4.8.10 GetIpcAdminMgmtPortStat

Description	Set the management port statistics.
	`STATUS GetIpcAdminMgmtPortStat(`
	`  IN int32 fnId,`
	`  OUT MgmtPortStat* mgmtPortStat);`

Applicability	IPC9xxx, IPC17xx
Example	rv=GetIpcAdminMgmtPortStat(0,&mgmtPortStat)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	mgmtPortStat	N/A	N/A

The management UART utilization in BC and master modes is been monitored in order not to exceed the maximal allowed utilization. There is no limit on the management UART utilization in slave mode. In case of exceeding the maximal utilization (BC and master modes), the device ignores the Get APIs and returns STATUS=-2 (No Service), indicates system busy. Set APIs are always excepted. The GetIpcAdminMgmtPortStat returns the management port statistics, MgmtPortStat.

5.4.9 BIST – Built-in Self Test

5.4.9.1 GetIpcAdminBistRepSlaveInit

Description	Get BIST result during initialization phase. Link is not required. This BIST executes ONLY once, during init.
	`STATUS GetIpcAdminBistRepSlaveInit(`
	`  IN int32 fnId,`
	`  OUT SlaveBistInit* slaveBistInit);`

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=GetIpcAdminBistRepSlaveInit(0,&slaveBistInit)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	slaveBistInit	N/A	0

5.4.9.2 GetIpcAdminBistRepSlaveShowtime

Description	Get BIST result during operation. Link is required.
	`STATUS GetIpcAdminBistRepSlaveShowtime(`
	`  IN int32 fnId,`
	`  OUT SlaveBistShowtime* slaveBistShowtime);`

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=GetIpcAdminBistRepSlaveShowtime(0,&slaveBistShowtime)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	slaveBistShowtime	N/A	0

5.4.9.3 SetIpcAdminBistPerRepSlaveEn

Description	Enables or disables an hourly periodic BIST report.
	`STATUS SetIpcAdminBistPerRepSlaveEn(`
	`  IN int32 fnId,`
	`  IN int32 repEn);`

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=SetIpcAdminBistPerRepSlaveEn(0,1)

Parameters	Parameter Name	Range		Default
	fnId	0		0
	repEn	0	Disable	0
		1	Enable	

5.4.9.4 GetIpcAdminBistPerRepSlaveEn

Description	Get the hourly periodic BIST report mode.
	`STATUS GetIpcAdminBistPerRepSlaveEn(`
	`  IN int32 fnId,`
	`  OUT int32* repEn);`

Applicability	IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Example	rv=GetIpcAdminBistPerRepSlaveEn(0,&repEn)

Parameters	Parameter Name	Range		Default
	fnId	0		0
	repEn	See	SetIpcAdminBistPerRepSlaveEn	0

5.4.9.5 GetIpcAdminBistRepMasterInit

Description	Get BIST result during initialization phase. Link is not required. This BIST is executed ONLY once, during init.
	`STATUS GetIpcAdminBistRepMasterInit(`
	`  IN int32 fnId,`
	`  OUT MasterBistInit* masterBistInit);`

Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode
Example	rv=GetIpcAdminBistRepMasterInit(0,&bistInit)

Parameters	Parameter Name	Range	Default

	fnId	0	0
	masterBistInit	N/A	0

5.4.9.6 GetIpcAdminBistRepMasterShowtime

Description	Get BIST result during operation. Link is required.
	`STATUS GetIpcAdminBistRepMasterShowtime(`
	`  IN int32 fnId,`
	`  IN int32 page,`
	`  OUT MasterBistShowtime* masterBistShowtime);`			

Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode			
Example	rv=GetIpcAdminBistRepMasterShowtime(0,2,&masterBistShowtime)			
Parameters	Parameter Name		Range	Default
	fnId	0		0
	page	0	CI 0-15	0
		1	CI 16-31	
		2	CI 32-47	
		3	CI 48-63	
	masterBistShowtime	N/A		0

5.4.9.7 SetIpcAdminBistPerRepMasterEn

Description	Enables or disables an hourly periodic BIST report.
	`STATUS SetIpcAdminBistPerRepMasterEn(`
	`  IN int32 fnId,`
	`  IN int32 repEn);`			

Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode			
Example	rv=SetIpcAdminBistPerRepMasterEn(0,1)			
Parameters	Parameter Name		Range	Default
	fnId	0		0
	repEn	0	Disable	1
		1	Enable	

5.4.9.8 GetIpcAdminBistPerRepMasterEn

Description	Get the hourly periodic BIST report mode.
	`STATUS GetIpcAdminBistPerRepMasterEn(`
	`  IN int32 fnId,`
	`  OUT int32* repEn);`			

Applicability	IPC9xxx – BC/Master modes, IPC17xx – Master mode			
Example	rv=GetIpcAdminBistPerRepMasterEn(0,&repEn)			
Parameters	Parameter Name		Range	Default
	fnId	0		0
	repEn	See	SetIpcAdminBistPerRepMasterEn	1

5.4.9.9 GetIpcAdminBistRepNativePll

Description	Get native PLL BIST results.
	`STATUS GetIpcAdminBistRepNativePll(`
	`  IN int32 fnId,`
	`  OUT BistNativePll* bistNativePll);`			

Applicability	IPC9xxx, IPC17xx			
Example	rv=GetIpcAdminBistRepNativePll(0,&bistNativePll)			
Parameters	Parameter Name		Range	Default
	fnId	0		0
	bistNativePll	N/A		N/A

Note *This API returns the current native PLL BIST results.*

5.4.9.10 GetIpcAdminBistRepPll

Description	Get PLL BIST results.

```
STATUS GetIpcAdminBistRepPll(
    IN int32 fnId,
    OUT BistPll* bistPll);
```

Applicability	IPC9xxx, IPC17xx
Example	rv=GetIpcAdminBistRepPll(0,&bistPll)

Parameters

Parameter Name	Range	Default
fnId	0	0
bistPll	N/A	N/A

Note *This API returns the current PLL BIST results.*

5.4.10 Signals Mux Selection

5.4.10.1 SetIpcAdminClkMuxASel

Description: Set the External Device Clock Mux A Selection.

```
STATUS SetIpcAdminClkMuxASel(
    IN int32 fnId,
    IN int32 clkMuxASel);
```

Applicability: IPC9600

Example: rv=SetIpcAdminClkMuxASel(0,1)

Parameters

Parameter Name	Range		Default
fnId	0		0
clkMuxASel	0	OSCIN 20MHz	5
	1	SYSIN x 2 (20MHz)	
	2	Reserved	
	3	Reserved	
	4	Reserved	
	5	Three state	

5.4.10.2 GetIpcAdminClkMuxASel

Description: Get the External Device Clock Mux A Selection.

```
STATUS GetIpcAdminClkMuxASel(
    IN  int32 fnId,
    OUT int32* clkMuxASel);
```

Applicability: IPC9600

Example: rv=GetIpcAdminClkMuxASel(0,&clkMuxASel)

Parameters

Parameter Name	Range		Default
fnId	0		0
clkMuxASel	See	SetIpcAdminClkMuxASel	5

5.4.10.3 GetIpcAdminClkMuxASel

Description: Get the External Device Clock Mux A Selection.

```
STATUS GetIpcAdminClkMuxASel(
    IN  int32 fnId,
    OUT int32* clkMuxASel);
```

Applicability: IPC9600

Example: rv=GetIpcAdminClkMuxASel(0,&clkMuxASel)

Parameters

Parameter Name	Range		Default
fnId	0		0
clkMuxASel	See	SetIpcAdminClkMuxASel	5

5.4.10.4 SetIpcAdminClkMuxBSel

Description: Set the External Device Clock Mux B Selection.

```
STATUS SetIpcAdminClkMuxBSel(
    IN int32 fnId,
    IN int32 clkMuxBSel);
```

Applicability	IPC9600			
Example	rv=SetIpcAdminClkMuxBSel(0,1)			
Parameters	Parameter Name		Range	Default
	fnId	0		0
	clkMuxBSel	0	Recovered 1PPS	5
		1	Recovered 1PPS with phase offs	
		2	CLKOUT1-Post PLL (10M)	
		3	Reserved	
		4	Reserved	
		5	Three state	

5.4.10.5 GetIpcAdminClkMuxBSel

Description	Get the External Device Clock Mux B Selection.			
	STATUS GetIpcAdminClkMuxBSel(**IN int32 fnId,** **OUT int32* clkMuxBSel);**			
Applicability	IPC9600			
Example	rv=GetIpcAdminClkMuxBSel(0,&clkMuxBSel)			
Parameters	Parameter Name		Range	Default
	fnId	0		0
	clkMuxBSel	See	SetIpcAdminClkMuxBSel	5

5.4.10.6 SetIpcAdminClkMuxCSel

Description	Set the External Device Clock Mux C Selection.			
	STATUS SetIpcAdminClkMuxCSel(**IN int32 fnId,** **IN int32 clkMuxCSel);**			
Applicability	IPC9600			
Example	rv=SetIpcAdminClkMuxCSel(0,0)			
Parameters	Parameter Name		Range	Default
	fnId	0		0
	clkMuxCSel	0	CLKIN	0
		1	CLKINS	

5.4.10.7 GetIpcAdminClkMuxCSel

Description	Get the External Device Clock Mux Selection.			
	STATUS GetIpcAdminClkMuxCSel(**IN int32 fnId,** **OUT int32* clkMuxCSel);**			
Applicability	IPC9600			
Example	rv=GetIpcAdminClkMuxCSel(0,&clkMuxCSel)			
Parameters	Parameter Name		Range	Default
	fnId	0		0
	clkMuxCSel	See	SetIpcAdminClkMuxCSel	0

5.4.10.8 SetIpcAdminClkMuxDSel

Description	Set the External Device Clock Mux D Selection.			
	STATUS SetIpcAdminClkMuxDSel(**IN int32 fnId,** **IN int32 clkMuxDSel);**			
Applicability	IPC9600			
Example	rv=SetIpcAdminClkMuxDSel(0,1)			
Parameters	Parameter Name		Range	Default
	fnId	0		0
	clkMuxDSel	0	CLKIN	0
		1	CLKINS	

5.4.10.9 GetIpcAdminClkMuxDSel

Description	Get the External Device Clock Mux Selection.
	`STATUS GetIpcAdminClkMuxDSel(`
	`IN int32 fnId,`
	`OUT int32* clkMuxDSel);`

Applicability	IPC9600
Example	rv=GetIpcAdminClkMuxDSel(0,&clkMuxDSel)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	clkMuxDSel	See SetIpcAdminClkMuxDSel	0

5.4.10.10 SetIpcAdminClkOutMuxSel

Description	Set the External Device Clock Out Mux Selection.
	`STATUS SetIpcAdminClkOutMuxSel(`
	`IN int32 fnId,`
	`IN int32 clkOutMuxSel);`

Applicability	IPC9600
Example	rv=SetIpcAdminClkOutMuxSel(0,1)

Parameters	Parameter Name	Range		Default
	fnId	0		0
	clkOutMuxSel	0	CLKOUT 10M	0
		1	CLKOUT 1.544/2.048M	
		4	CLKC	
		5	ETHRXCLKO1 25/125M	
		8	Three state	

5.4.10.11 GetIpcAdminClkOutMuxSel

Description	Get the External Device Clock Out Mux Selection.
	`STATUS GetIpcAdminClkOutMuxSel(`
	`IN int32 fnId,`
	`OUT int32* clkOutMuxSel);`

Applicability	IPC9600
Example	rv=GetIpcAdminClkOutMuxSel(0,&clkOutMuxSel)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	clkOutMuxSel	See SetIpcAdminClkOutMuxSel	0

5.4.10.12 SetIpcAdminClkRefMuxSel

Description	Set the ETHREFCLK (Ethernet Reference Clock) out Mux Selection.
	`STATUS SetIpcAdminClkRefMuxSel(`
	`IN int32 fnId,`
	`IN int32 clkRefMuxSel);`

Applicability	IPC9600
Example	rv=SetIpcAdminClkRefMuxSel(0,1)

Parameters	Parameter Name	Range		Default
	fnId	0		0
	clkRefMuxSel	0	Internal 125MHz driven by OSCIN	0
		1	ETHTXCLKIN 125MHz	
		3	Three state	

5.4.10.13 GetIpcAdminClkRefMuxSel

Description	Get the ETHREFCLK (Ethernet Reference Clock) out Mux Selection.
	`STATUS GetIpcAdminClkRefMuxSel(`
	`IN int32 fnId,`
	`OUT int32* clkRefMuxSel);`

Applicability	IPC9600
Example	rv=GetIpcAdminClkRefMuxSel(0,&clkRefMuxSel)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	clkRefMuxSel	See SetIpcAdminClkRefMuxSel	0

5.4.10.14 SetIpcAdminClkSysInMuxSel

Description	Set the SYSIN Mux Selection.
	`STATUS SetIpcAdminClkSysInMuxSel(`
	`IN int32 fnId,`
	`IN int32 clkSysInMuxSel);`

Applicability	IPC9xxx – BC/Slave modes, IPC17xx Slave mode
Example	rv=SetIpcAdminClkSysInMuxSel(0,0)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	clkSysInMuxSel	0 SYSIN 1 CLKIN	0

5.4.10.15 GetIpcAdminClkSysInMuxSel

Description	Get the SYSIN Mux Selection.
	`STATUS GetIpcAdminClkSysInMuxSel(`
	`IN int32 fnId,`
	`OUT int32* clkSysInMuxSel);`

Applicability	IPC9xxx – BC/Slave modes, IPC17xx Slave mode
Example	rv=GetIpcAdminClkSysInMuxSel(0,&clkSysInMuxSel)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	clkSysInMuxSel	See SetIpcAdminClkSysInMuxSel	0

5.4.11 Assisted Partial Timing Support – APTS

5.4.11.1 General

The Assisted Partial Timing Support (APTS) enhanced the synchronization performance. The APTS is applicable to IPC9xxx products only. The APTS consist the external time input interface (ETI). The inclusion of an ETI in BC and T-BC, enables to use external 1PPS clock source (e.g. from GPS) as one of the BMC sources for selection or to use it for Asymmetry Compensation (ASC). The external 1PPS clock source shall be connected to CLKIN pin.
The ASC operates in BC and slave modes. ASC enables the device to use external 1PPS clock source (e.g. from GPS) for asymmetry compensation. The ASC usually performs its operation within up to 15 minutes. The ASC operates only in case the device is in Trace or Lock states. The ascInfo structure provides the operational ASC information. The ascInfo is also presented by the device log message 0170. The AsymCompClr enable to clear the ASC information and actions.

The aptsMode control the device operation as follow:

- aptsMode=0 – Disabled.
- aptsMode=1 – Auto ETI/PTP Master. The device monitors the ETI port in addition to the PTP masters. The BMC automatically selects the best PTP master, compared it to the ETI master and select the best out of those two. In case ETI was selected, the device operates simultaneously with the PTP master as the ETI master while the device clock is been locked to the ETI master.
- aptsMode=2 – Manual switch to PTP Master. The BMC automatically selects the best PTP master, without considering ETI. ETI entry is included in bmcKnownMasterTable.
- aptsMode=3 – Manual switch to ETI. The device select the ETI. In addition BMC automatically selects the best PTP master but is not locked to. ETI entry is included in bmcKnownMasterTable. parentDS (GetIpcPtpDsParent) and Announce packets presents/includes the PTP master information (and not ETI information).

The entire virtualPtpPort dataset as set by SetIpcPtpAptsVppPar API enables to set the properties of the ETI. Those properties shall be used by the BMC for the master selection. The virtualPtpPort include the following members: clockClassMode, clockClassConf, clockAccuracy, offsetScaledLogVar, portNum and localPriority.

In case of BC mode (startMode=4) the selection is been done as defined by IEEE 1588 v2.
In case of T-BC mode (startMode=24) the selection is been done as defined by G.8275.1 Annex C.

For proper operation the following shall be noted:
The virtualPtpPort.localPriority shall not be modified by the user. It shall be kept on its default value (=128).
The PortDs.localPriority shall not be modified by the user. It shall be kept on its default value (=128)

virtualPtpPort doesn't contain Priority1 and Priority2. Priority1 and Priority2 doesn't used for BMC comparison. The ETI entry as presented by GetIpcPtpBmcKnownMasterTable presents Priority1 = 0, Priority2 = 0.

5.4.11.2 SetIpcPtpAptsMode

Description	Set the APTS mode.
	`STATUS SetIpcPtpAptsMode(`
	`  IN int32 fnId,`
	`  IN int32 aptsMode);`

Applicability	IPC9xxx – BC/Slave modes
Example	rv=SetIpcPtpAptsMode(0,1)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	aptsMode	0 – Off 1 – Auto ETI/PTP Master 2 – Manual switch to PTP Master 3 – Manual switch to ETI	0

Applicability:
-- All AptsMode applicable for BC mode

5.4.11.3 GetIpcPtpAptsMode

Description	Get the APTS mode.
	`STATUS GetIpcPtpAptsMode(`
	`  IN int32 fnId,`
	`  OUT int32* aptsMode);`

Applicability	IPC9xxx – BC/Slave modes
Example	rv=GetIpcPtpAptsMode(0,&aptsMode)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	aptsMode	See SetIpcPtpAptsMode	0

5.4.11.4 SetIpcPtpAptsVppPar

Description	Set the virtual PTP port (Vpp) parameters.
	`STATUS SetIpcPtpAptsVppPar(`
	`  IN int32 fnId,`
	`  IN int32 reserved1,`
	`  IN int32 clockClassMode,`
	`  IN int32 clockClassConf,`
	`  IN int32 clockClass,`
	`  IN int32 clockAccuracy,`
	`  IN int32 offsetScaledLogVar,`
	`  IN int32 portNum,`
	`  IN int32 localPriority);`

Applicability	IPC9xxx – BC mode
Example	rv=SetIpcPtpAptsVppPar(0,-1,0,248,-1,254,65535,256,128) Set clockAccuracy only: rv=SetIpcPtpAptsVppPar(0,-1,-1,-1,-1,100,-1,-1,-1)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	Reserved1	-1	-1

	clockClassMode	0 – Manual: clockClass controlled by clockClassConf	0
		1 – Auto: clockClass controlled by PLL. VPP	
		2 – Auto: clockClass controlled by PLL	
	clockClassConf	0 – 255	248
	clockClass	0 – 255	-1
	clockAccuracy	0 – 255	254
	offsetScaledLogVar	0 – 65535	65535
	portNum	0 – 65535	256
	localPriority	0 – 255	128

Note *Using -1 shall cause the device to ignore the parameter and leave it unchanged.*

Note *clockClass is a result of clockClassMode, clockClassConf and GState.state as reported by GetIpcPtpState. It is applicable only for GetIpcPtpAptsVppPar only.*
clockClass can't be controlled by SetIpcPtpAptsVppPar.

Note *localPriority applicable for starMode=24 only (ptpBmcMode=10). In other cases it shall be set as 1*

5.4.11.5 GetIpcPtpAptsVppPar

Description	Get the virtual PTP port (Vpp) parameters.
	`STATUS GetIpcPtpAptsVppPar(`
	`  IN int32 fnId,`
	`  OUT virtualPtpPort* virtualPtpPort);`		

Applicability	IPC9xxx – BC mode		
Example	rv=GetIpcPtpAptsVppPar(0,&virtualPtpPort)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	virtualPtpPort	See SetIpcPtpAptsVppPar	N/A

5.4.11.6 GetIpcPtpAptsState

Description	Get the the APTS state.
	`STATUS GetIpcPtpAptsState(`
	`  IN int32 fnId,`
	`  OUT int32* aptsState);`		

Applicability	IPC9xxx – BC mode		
Example	rv=GetIpcPtpAptsState(0,&aptsState)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	aptsState	0 PTP master 1 ETI master	0

5.4.11.7 GetIpcPtpAptsAscInfo

Description	Get the Asymmetry Compensation Information.
	AscInfo.Cnt - Number of time-updates by ASC
	AscInfo.LastCorr - Last asymmetry compensation correction in nsec
	AscInfo.AggCorr - Aggregated asymmetry compensation correction in nsec
	AscInfo.TotalCorr - Total asymmetry compensation correction in nsec
	`STATUS GetIpcPtpAptsAscInfo(`
	`  IN int32 fnId,`
	`  OUT AscInfo* ascInfo);`		

Applicability	IPC9xxx – BC/Slave modes		
Example	rv=GetIpcPtpAptsAscInfo(0,&ascInfo)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	ascInfo	N/A	N/A

5.4.11.8 SetIpcPtpAptsAscClr

Description	Clear the Asymmetry Compensation counter and last correction value.
	`STATUS SetIpcPtpAptsAscClr(`
	`  IN  int32 fnId`
	`  IN  int32 AsymCompClr);`

Applicability	IPC9xxx – BC/Slave modes
Example	rv=SetIpcPtpAptsAscClr(0,2)

Parameters

Parameter Name	Range	Default
fnId	0	0
AsymCompClr	0 – Clear *AscInfo.Cnt* and *AscInfo.LastCorr* only. *AscInfo.LastCorr* already been used. This action clears information only.	0
	1 – Clear aggregated asymmetry correction. In addition to performing clear as been defined by *AsymCompClr*=0, it clears *AscInfo.AggCorr*. This action clears the entire aggregated asymmetry correction from the time the ASC was enabled.	
	2 – Clear all history correction. In addition to performing clear as been defined by *AsymCompClr*=1, it clears *AscInfo.TotalCorr*. This action clears the entire *corrAsymmetry* as been controlled by SetIpcAdminPpsOutAsymmetryCorr API	

5.4.12 External Device

5.4.12.1 SetIpcAdminExtDeviceReset

Description	Resets the external device.
	`STATUS SetIpcAdminExtDeviceReset(`
	`  IN int32 fnId`
	`  IN int32 extDevID`
	`  IN int32 extDevRstMode);`

Applicability	IPC9600
Example	rv=SetIpcAdminExtDeviceReset(0,0,0)

Parameters

Parameter Name	Range	Default
fnId	0	0
extDevID	0 – External Device A 1 – External Device B 2 – External Device A + B	N/A
extDevRstMode	0 – Reset 1 – Set Reset pin to high (1) 2 – Set Reset pin to low (0)	N/A

Note *extDevRstMode=0: do extDevRstMode=1 for 1 sec then extDevRstMode=0.*

Note *Reset can be call only after the previous reset was completed.*

5.4.12.2 GetIpcAdminExtDeviceStatus

Description	Get the External Device Status.
	`STATUS GetIpcAdminExtDeviceStatus(`
	`  IN  int32 fnId,`
	`  OUT int32* ED_ST);`

Applicability	IPC9600
Example	rv=GetIpcAdminExtDeviceStatus(0,&EDStatus.ED_ST)

Parameters

Parameter Name	Range	Default
fnId	0	0

	EDStatus.ED_ST	N/A	N/A

5.4.12.3 GetIpcAdminExtDeviceStatusB

Description	Get the External Device Status, device B.
	`STATUS GetIpcAdminExtDeviceStatusB(`
	`  IN  int32 fnId,`
	`  OUT int32* ED_STB);`		

Applicability	IPC9600		
Example	rv=GetIpcAdminExtDeviceStatusB(0,&EDStatus.ED_STB)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	EDStatus.ED_STB	N/A	N/A

5.4.12.4 GetIpcAdminExtDeviceStatusTime

Description	Get the External Device Status / Global Status time and elapsed time.
	`STATUS GetIpcAdminExtDeviceStatusTime(`
	`  IN  int32 fnId,`
	`  OUT EDSTT* EDSTT);`		

Applicability	IPC9600 – BC/Slave modes		
Example	rv=GetIpcAdminExtDeviceStatusTime(0,&eDSTT)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	eDSTT	N/A	N/A

5.4.12.5 GetIpcAdminExtDeviceGlobStatus

Description	Get the External Device Global Status.
	`STATUS GetIpcAdminExtDeviceGlobStatus(`
	`  IN  int32 fnId,`
	`  OUT int32* G_ST);`		

Applicability	IPC9600		
Example	rv=GetIpcAdminExtDeviceGlobStatus(0,&EDStatus.G_ST)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	EDStatus.G_ST	N/A	N/A

5.4.12.6 GetIpcAdminExtDeviceStatusEx

Description	Get the External Device Status Extended.
	`STATUS GetIpcAdminExtDeviceStatusEx(`
	`  IN  int32 fnId,`
	`  OUT EDStatus* eDStatus);`		

Applicability	IPC9600		
Example	rv=GetIpcAdminExtDeviceStatusEx(0,&eDStatus)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	eDStatus	N/A	N/A

5.4.12.7 SetIpcAdminExtDeviceSpi

Description	Set the External Device SPI.
	`STATUS SetIpcAdminExtDeviceSpi(`
	`  IN  int32 fnId,`
	`  IN  int32 edSpi);`		

Applicability	IPC9600		
Example	rv=SetIpcAdminExtDeviceSpi(0,1)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	edSpi	0 – External device A 1 – External device B	0

5.4.12.8 GetIpcAdminExtDeviceSpi

Description	Get the External Device SPI.
	`STATUS GetIpcAdminExtDeviceSpi(`
	`  IN  int32 fnId,`
	`  OUT int32* edSpi);`

Applicability	IPC9600
Example	rv=GetIpcAdminExtDeviceSpi(0,&edSpi)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	edSpi	See SetIpcAdminExtDeviceSpi	0

5.4.12.9 SetIpcAdminExtDeviceSmBp

Description	Set the External Device state machine bypass during device initialization. For development purposes only.
	`STATUS SetIpcAdminExtDeviceSmBp(`
	`  IN  int32 fnId,`
	`  IN  int32 smBp);`

Applicability	IPC9600
Example	rv=SetIpcAdminExtDeviceSmBp(0,1)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	smBp	0 – Normal 1 – Bypass	0

(1) This API is applicable only if FLASH memory is used.
(2) The new parameters settings will take effect only after storing configDB in FLASH memory and device reset.

In order to manually control the SM and to get maximal SM information into the log:
SetIpcAdminExtDeviceSmBp(0,1)
SetIpcAdminExtDeviceRep(0,1,1,0,0,0,1,1)
SetIpcAdminStore
SetIpcPtpAdminStop
SetIpcPtpAdminStart
Or
SetIpcAdminExtDeviceSmBp(0,1)
SetIpcAdminExtDeviceRep(0,1,1,0,0,0,1,1)
SetIpcAdminStore
SetIpcPtpAdminReset

5.4.12.10 GetIpcAdminExtDeviceSmBp

Description	Get the External Device state machine bypass during device initialization.
	`STATUS GetIpcAdminExtDeviceSmBp(`
	`  IN  int32 fnId,`
	`  OUT int32* smBp);`

Applicability	IPC9600
Example	rv=GetIpcAdminExtDeviceSmBp(0,&smBp)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	smBp	See SetIpcAdminExtDeviceSmBp	0

5.4.12.11 GetIpcAdiProdId

Description	Get product ID.
	`STATUS GetIpcAdiProdId(`
	`  IN int32 fnId,`
	`  OUT int32* prodId);`

Applicability	IPC9600
Example	rv=GetIpcAdiProdId(0,&prodId)

Parameters	Parameter Name	Range	Default
	fnId	0	0
	prodId	0x06	0x06

5.4.12.12 SetIpcAdiSysClkFreq

Description	Set the system clock frequency.
	`STATUS SetIpcAdiSysClkFreq(`
	`  IN int32 fnId,`
	`  IN int32 adiSysClkFreq);`		

Applicability	IPC9600		
Example	rv=SetIpcAdiSysClkFreq(0,adiSysClkFreq)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	adiSysClkFreq	0 – 20MHz	0

5.4.12.13 GetIpcAdiSysClkFreq

Description	Get the system clock frequency.
	`STATUS GetIpcAdiSysClkFreq(`
	`  IN int32 fnId,`
	`  OUT int32* adiSysClkFreq);`		

Applicability	IPC9600		
Example	rv=GetIpcAdi(0,&adiSysClkFreq)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	adiSysClkFreq	See SetIpcAdi	0

5.4.12.14 SetIpcAdiDpllMode

Description	Set the DPLL / DDS mode.
	`STATUS SetIpcAdiDpllMode(`
	`  IN int32 fnId,`
	`  IN int32 adiDpllMode);`		

Applicability	IPC9600		
Example	rv=SetIpcAdiDpllMode(0,1)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	adiDpllMode	0 – DDS 1 – PLL	0

5.4.12.15 GetIpcAdiDpllMode

Description	Get the DPLL / DDS mode.
	`STATUS GetIpcAdiDpllMode(`
	`  IN int32 fnId,`
	`  OUT int32* adiDpllMode);`		

Applicability	IPC9600		
Example	rv=GetIpcAdiDpllMode(0,&adiDpllMode)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	adiDpllMode	0 – See SetIpcAdiDpllMode	0

5.4.12.16 SetIpcAdiClkOutEn

Description	Set the clock output enable.
	`STATUS SetIpcAdiClkOutEn(`
	`  IN int32 fnId,`
	`  IN int32 adiPortNum,`
	`  IN int32 adiPN,`
	`  IN int32 adiClkOutEn);`		

Applicability	IPC9600		
Example	rv=SetIpcAdiClkOutEn(0,0,0,1)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	adiPortNum	0 – OUT0	0

	1 – OUT1 2 – OUT2 3 – OUT3	
adiPN	0 – Positive 1 – Negative	0
adiClkOutEn	0 – Disable 1 – Enable	0

5.4.12.17 GetIpcAdiClkOutEn

Description: Get the clock output enable.

```
STATUS GetIpcAdiClkOutEn(
    IN int32 fnId,
    IN int32 adiPortNum,
    IN int32 adiPN,
    OUT int32* adiClkOutEn);
```

Applicability: IPC9600

Example: rv=GetIpcAdiClkOutEn(0,0,0,&adiClkOutEn)

Parameters:

Parameter Name	Range	Default
fnId	0	0
adiPortNum	See SetIpcAdiClkOutEn	N/A
adiPN	See SetIpcAdiClkOutEn	N/A
adiClkOutEn	See SetIpcAdiClkOutEn	0

5.4.12.18 SetIpcAdiDdsProfile

Description: Set the DDS profile.

```
STATUS SetIpcAdiDdsProfile(
    IN int32 fnId,
    IN int32 adiDdsProfile);
```

Applicability: IPC9600

Example: rv=SetIpcAdiDdsProfile(0,1)

Parameters:

Parameter Name	Range	Default
fnId	0	0
adiDdsProfile	0 – SyncE profile. 250 MHz fdds. Outputs: 1Hz,8KHz,10MHz,25MHz,125MHz	0
	1 – T1 profile. 200.72 MHz fdds. Outputs: 1Hz,8KHz,1.544MHz	
	2 – E1 profile. 204.8 MHz fdds. Outputs: 1Hz,8KHz,2.048MHz,12.8MHz	
	3 – Profile 3. 200 MHz fdds. Outputs: 1Hz,8KHz,10MHz,20MHz,25MHz	

5.4.12.19 GetIpcAdiDdsProfile

Description: Get the DDS profile.

```
STATUS GetIpcAdiDdsProfile(
    IN int32 fnId,
    OUT int32* adiDdsProfile);
```

Applicability: IPC9600

Example: rv=GetIpcAdiDdsProfile(0,&adiDdsProfile)

Parameters:

Parameter Name	Range	Default
fnId	0	0
adiDdsProfile	0 – 3	0

5.4.12.20 SetIpcAdiClkOutFreq

Description: Set the clock output frequency.

```
STATUS SetIpcAdiClkOutFreq(
    IN int32 fnId,
    IN int32 adiPortNum,
    IN int32 adiClkOutFreq);
```

Applicability	IPC9600		
Example	rv=SetIpcAdiClkOutFreq(0,0,0)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	adiPortNum	0 – OUT0 1 – OUT1 2 – OUT2 3 – OUT3	0
	adiClkOutFreq	0 – 1PPS 1 – 8K 2 – 1.5M 3 – 2M 4 – 10M 5 – 12.8M 6 – 20M 7 – 25M 8 – 125M	0

5.4.12.21 GetIpcAdiClkOutFreq

Description	Get the clock output frequency.

```
STATUS GetIpcAdiClkOutFreq(
    IN int32 fnId,
    IN int32 adiPortNum,
    OUT int32* adiClkOutFreq);
```

Applicability	IPC9600		
Example	rv=GetIpcAdiClkOutFreq(0,0,&adiClkOutFreq) In case the returned adiClkOutFreq=4 – the OUT0 pin clock is 10MHz		
Parameters	Parameter Name	Range	Default
	fnId	0	N/A
	adiPortNum	See SetIpcAdiClkOutFreq	N/A
	adiClkOutFreq	See SetIpcAdiClkOutFreq	0

5.4.12.22 SetIpcAdiClkInEn

Description	Set the clock input enable.

```
STATUS SetIpcAdiClkInEn(
    IN int32 fnId,
    IN int32 adiPortNum,
    IN int32 adiPN,
    IN int32 adiClkInEn);
```

Applicability	IPC9600		
Example	rv=SetIpcAdiClkInEn(0,0,0,1)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	adiPortNum	0 – REFA/AA 1 – REFB/BB 2 – REFC/CC 3 – REFD/DD	0
	adiPN	0 – Positive (e.g. REFA) 1 – Negative (e.g. REFAA)	0
	adiClkInEn	0 – Disable 1 – Enable	0

Note

As examples:
REFA: adiPortNum=0, adiPN=0.
REFAA: adiPortNum=0, adiPN=1.
REFB: adiPortNum=1, adiPN=0.
REFBB: adiPortNum=1, adiPN=1.

5.4.12.23 GetIpcAdiClkInEn

Description	Get the clock input enable.

```
STATUS GetIpcAdiClkInEn(
    IN int32 fnId,
    IN int32 adiPortNum,
    IN int32 adiPN,
    OUT int32* adiClkInEn);
```

Applicability	IPC9600
Example	rv=GetIpcAdiClkInEn(0,0,0,&adiClkInEn)

Parameters

Parameter Name	Range	Default
fnId	0	0
adiPortNum	See SetIpcAdiClkInEn	N/A
adiPN	See SetIpcAdiClkInEn	N/A
adiClkInEn	See SetIpcAdiClkInEn	0

5.4.12.24 SetIpcAdiClkInFreq

Description: Set the clock input frequency.

```
STATUS SetIpcAdiClkInFreq(
    IN int32 fnId,
    IN int32 adiPortNum,
    IN int32 adiPN,
    IN int32 adiClkInFreq);
```

Applicability	IPC9600
Example	rv=SetIpcAdiClkInFreq(0,0,0,0)

Parameters

Parameter Name	Range	Default
fnId	0	0
adiPortNum	0 – REFA/AA 1 – REFB/BB 2 – REFC/CC 3 – REFD/DD	0
adiPN	0 – Positive (e.g. REFA) 1 – Negative (e.g. REFAA)	0
adiClkInFreq	0 – 1PPS 1 – 8K 2 – 1.5M 3 – 2M 4 – 10M 5 – 12.8M 6 – 20M 7 – 25M 8 – 125M	0

5.4.12.25 GetIpcAdiClkInFreq

Description: Get the clock input frequency.

```
STATUS GetIpcAdiClkInFreq(
    IN int32 fnId,
    IN int32 adiPortNum,
    OUT int32* adiClkOutFreq);
```

Applicability	IPC9600
Example	rv=GetIpcAdiClkInFreq(0,0,0,&adiClkInFreq)

Parameters

Parameter Name	Range	Default
fnId	0	0
adiPortNum	See SetIpcAdiClkInFreq	N/A
adiPN	See SetIpcAdiClkInFreq	N/A
adiClkInFreq	See SetIpcAdiClkInFreq	0

5.4.12.26 SetIpcAdiClkOutFreqOfstRaw

Description: Set the frequency offset in ppb (10^-9) for DDS mode.

```
STATUS SetIpcAdiClkOutFreqOfstRaw(
    IN int32 fnId,
    IN int32 adiClkOutFreqOfst);
```

Applicability	IPC9600		
Example	rv=SetIpcAdiClkOutFreqOfstRaw(0,1000)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	adiClkOutFreqOfst	+/-100000000 (100000ppm)	0

Note *adiClkOutFreqOfst max value +/-100000000 (~= +/-100000ppm).*

Note *Original spec: adiClkOutFreqOfst 28bits, max value +/-2^27=+/-268435456 (~= +/-268000ppm).*

5.4.12.27 GetIpcAdiClkOutFreqOfstRaw

Description: Get the the frequency offset in ppb (10^-9) for DDS mode.

```
STATUS GetIpcAdiClkOutFreqOfstRaw(
    IN int32 fnId,
    OUT int32* adiClkOutFreqOfst);
```

Applicability	IPC9600		
Example	rv=GetIpcAdiClkOutFreqOfstRaw(0,&adiClkOutFreqOfst)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	adiClkOutFreqOfst	See SetIpcAdiClkOutFreqOfstRaw	0

5.4.12.28 SetIpcAdiClkOutFreqOfst

Description: Set the frequency offset in ppt (10^-12) for DDS mode.

```
STATUS SetIpcAdiClkOutFreqOfst(
    IN int32 fnId,
    IN int32 adiClkOutFreqOfst);
```

Applicability	IPC9600		
Example	rv=SetIpcAdiClkOutFreqOfst(0,1000)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	adiClkOutFreqOfst	+/-2147483648 (2147ppm)	0

Note *adiClkOutFreqOfst in ppt, 10^-12. 32bits, max value +/-2^31=+/-2147483648 (~= +/-2147).*

5.4.12.29 GetIpcAdiClkOutFreqOfst

Description: Get the the frequency offset in ppt (10^-12) for DDS mode.

```
STATUS GetIpcAdiClkOutFreqOfst(
    IN int32 fnId,
    OUT int32* adiClkOutFreqOfst);
```

Applicability	IPC9600		
Example	rv=GetIpcAdiClkOutFreqOfst(0,&adiClkOutFreqOfst)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	adiClkOutFreqOfst	See SetIpcAdiClkOutFreqOfst	0

5.4.12.30 SetIpcAdiClkOutFreqOfstDpll

Description: Set the frequency offset in ppt (10^-12) for DPLL mode.

```
STATUS SetIpcAdiClkOutFreqOfstDpll(
    IN int32 fnId,
    IN int32 adiClkOutFreqOfst);
```

Applicability	IPC9600		
Example	rv=SetIpcAdiClkOutFreqOfstDpll(0,1000)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	adiClkOutFreqOfst	+/-2147483648 (2147ppm)	0

Note *adiClkOutFreqOfst in ppt, 10^-12. 28bits, max value +/-2^27=+/-134217728. (~= +/-134).*

5.4.12.31 GetIpcAdiClkOutFreqOfstDpll

Description	Get the the frequency offset in ppt (10^-12) for DPLL mode.

```
STATUS GetIpcAdiClkOutFreqOfstDpll(
    IN int32 fnId,
    OUT int32* adiClkOutFreqOfst);
```

Applicability	IPC9600
Example	rv=GetIpcAdiClkOutFreqOfstDpll(0,&adiClkOutFreqOfst)

Parameters

Parameter Name	Range	Default
fnId	0	0
adiClkOutFreqOfst	See SetIpcAdiClkOutFreqOfstDpll	0

5.4.12.32 SetIpcAdiLoopBw

Description	Set the loop bandwidth.

```
STATUS SetIpcAdiLoopBw(
    IN int32 fnId,
    IN int32 adiPortNum,
    IN int32 adiPN,
    IN int32 adiLoopBw);
```

Applicability	IPC9600
Example	rv=SetIpcAdiLoopBw(0,2,0)

Parameters

Parameter Name	Range	Default
fnId	0	0
adiPortNum	0 – REFA/AA 1 – REFB/BB 2 – REFC/CC 3 – REFD/DD	0
adiPN	0 – Positive 1 – Negative	0
adiLoopBw	0 – 1mHz 1 – 2mHz 2 – 4mHz 3 – 8mHz 4 – 16mHz 5 – 32mHz 6 – 64mHz 7 – 128mHz 8 – 256mHz 9 – 512mHz 10–1.024Hz	10

As examples:
REFA: adiPortNum=0, adiPN=0.
REFAA: adiPortNum=0, adiPN=1.
REFB: adiPortNum=1, adiPN=0.
REFBB: adiPortNum=1, adiPN=1.

5.4.12.33 GetIpcAdiLoopBw

Description	Get the loop bandwidth.

```
STATUS GetIpcAdiLoopBw(
    IN int32 fnId,
    IN int32 adiPortNum,
    IN int32 adiPN,
    OUT int32* adiLoopBw);
```

Applicability	IPC9600
Example	rv=GetIpcAdiLoopBw(0,0,0,&adiLoopBw)

Parameters

Parameter Name	Range	Default
fnId	0	0
adiPortNum	See SetIpcAdiLoopBw	N/A
adiPN	See SetIpcAdiLoopBw	N/A
adiLoopBw	See SetIpcAdiLoopBw	10

5.4.12.34 SetIpcAdiLfCoef

Description	Set the Loop filter coefficients.

```
STATUS SetIpcAdiLfCoef(
    IN int32 fnId,
    IN int32 adiPortNum,
    IN int32 adiPN,
    IN int32 adiClkInFreq,
    IN int32 adiLoopBw);
```

Applicability	IPC9600
Example	rv=SetIpcAdiLfCoef(0,0,0,0,0)

Parameters

Parameter Name	Range	Default
fnId	0	0
adiPortNum	0 – REFA/AA 1 – REFB/BB 2 – REFC/CC 3 – REFD/DD	N/A
adiPN	0 – Positive 1 – Negative	N/A
adiClkInFreq	0 – 1PPS 1 – 8K 2 – 1.5M 3 – 2M 4 – 10M 5 – 12.8M 6 – 20M 7 – 25M 8 – 125M	N/A
adiLoopBw	0 – 1mHz 1 – 2mHz 2 – 4mHz 3 – 8mHz 4 – 16mHz 5 – 32mHz 6 – 64mHz 7 – 128mHz 8 – 256mHz 9 – 512mHz 10–1.024Hz	N/A

5.4.12.35 SetIpcAdiReg

Description	Set register.

```
STATUS SetIpcAdiReg(
    IN int32 fnId,
    IN int32 adiAddr,
    IN u_char adiReg);
```

Applicability	IPC9600
Example	rv=SetIpcAdiReg(0,0,5)

Parameters

Parameter Name	Range	Default
fnId	0	0
adiAddr	N/A	N/A
adiReg	N/A	N/A

5.4.12.36 GetIpcAdiReg

Description	Get register.

```
STATUS GetIpcAdiReg(
    IN int32 fnId,
    IN int32 adiAddr,
    OUT u_char adiReg);
```

Applicability	IPC9600

Example	rv=GetIpcAdiReg(0,0,&adiReg)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	adiAddr	N/A	N/A
	adiReg	N/A	N/A

5.4.12.37 SetIpcAdiUpdate

Description	Set update.

```
STATUS SetIpcAdiUpdate(
    IN int32 fnId,
    void);
```

Applicability	IPC9600		
Example	rv=SetIpcAdiUpdate(0)		
Parameters	Parameter Name	Range	Default
	fnId	0	0

5.4.12.38 GetIpcAdiStatus

Description	Get the device status.

```
STATUS GetIpcAdiStatus(
    IN int32 fnId,
    OUT AdiState* adiState);
```

Applicability	IPC9600		
Example	rv=GetIpcAdiStatus(0,&adiState)		
Parameters	Parameter Name	Range	Default
	fnId	0	0
	adiState.adiRefClkState	8 bits [0] – Ref A valid / Ref A invalid Bar [1] – Ref AA valid / Ref AA invalid Bar [2] – Ref B valid / Ref B invalid Bar [3] – Ref BB valid / Ref BB invalid Bar [4] – Ref C valid / Ref C invalid Bar [5] – Ref CC valid / Ref CC invalid Bar [6] – Ref D valid / Ref D invalid Bar [7] – Ref DD valid / Ref DD invalid Bar	N/A
	adiState.adiDpllState	0 – Freerunning 1 – Tracking 2 – Frequency locked 3 – Phase locked 4 – Phase & frequency locked 5 – Holdover 6 – Switching -1– Error	N/A

Note

The adiState.adiRefClkState is decimal representation of the binary value.
As examples:
Ref A valid, all the rest invalid - adiState.adiRefClkState=1
Ref A valid, Ref B valid, all the rest invalid - adiState.adiRefClkState=3

5.4.12.39 SetIpcAdiDistSync

Description	Enables the sync mask for all outputs.

```
STATUS SetIpcAdiDistSync(
    IN int32 fnId);
```

Applicability	IPC9600		
Example	rv=SetIpcAdiDistSync(0)		
Parameters	Parameter Name	Range	Default
	fnId	0	0

5.4.12.40 SetIpcAdiStateMachineMode

Description	Set the state machine mode.
	`STATUS SetIpcAdiStateMachineMode(`
	`  IN int32 fnId,`
	`  IN int32 adiStateMacMode);`

Applicability	IPC9600
Example	rv=SetIpcAdiStateMachineMode(0,1)

Parameters

Parameter Name	Range	Default
fnId	0	0
adiStateMacMode	0 – Disable 1 – Enable 2 – Enable & Report in case msg 0507 content changed 3 – Enable & Report in case msg 0507 content changed & SetIpcAdiReg 4 – Enable & Report in case msg 0507 content changed & SetIpcAdiReg & GetIpcAdiReg 5 – Enable & Report in case msg 0507 content changed & SetIpcAdiReg & GetIpcAdiReg & additional SM information	0

Note *Report: Msg0507 SM: rv S adiSmStatus F fpo[0] fpo[1] fpo[2] fpo[3] fpo[4] R ref_period[0] ref_period[1] ref_period[2] ref_period[3] ref_period[4] ref_period[5] ref_period[6] ref_period[7] (e.g. SM: 0 S 0 V 0 0 0 0 0 R 0 0 0 0 0 0 0 0)*

5.4.12.41 GetIpcAdiStateMachineMode

Description	Get the state machine mode.
	`STATUS GetIpcAdiStateMachineMode(`
	`  IN int32 fnId,`
	`  OUT int32* adiStateMecMode);`

Applicability	IPC9600
Example	rv=GetIpcAdiStateMachineMode(0,&adiStateMacMode)

Parameters

Parameter Name	Range	Default
fnId	0	0
adiStateMacMode	See SetIpcAdiStateMachineMode	0

5.4.12.42 GetIpcAdiStateMachineStatus

Description	Get the state machine status invokes the AdiStateMachine and returns its status.
	`STATUS GetIpcAdiStateMachineStatus(`
	`  IN int32 fnId,`
	`  IN int32 adiShowOutputClock,`
	`  OUT int32* adiSmStatus);`

Applicability	IPC9600
Example	rv=GetIpcAdiStateMachineStatus(0,adiShowOutputClock,&adiSmStatus)

Parameters

Parameter Name	Range	Default
fnId	0	0
adiShowOutputClock	N/A	N/A
adiSmStatus	N/A	N/A

5.4.12.43 SetIpcAdiStateMachineReset

Description	Reset state machine.
	`STATUS SetIpcAdiStateMachineReset(`
	`  IN int32 fnId,`
	`  void);`

Applicability	IPC9600
Example	rv=SetIpcAdiStateMachineReset(0)

Parameters

Parameter Name	Range	Default
fnId	0	0

5.4.13 External Device 4 – High Level Configuration

The following APIs are for information only. They are embedded in the SetIpcPtpAdminStart API and shall not be call directly.

5.4.13.1 SetIpcAdiConf0

Description	Set configuration 0.
	`STATUS SetIpcAdiConf0(`
	`  IN int32 fnId);`		

Applicability	IPC9600		
Example	rv=SetIpcAdiConf0(0)		
Parameters	Parameter Name	Range	Default
	fnId	0	0

5.4.13.2 SetIpcAdiConfA

Description	Set configuration A.
	`STATUS SetIpcAdiConfA(`
	`  IN int32 fnId);`		

Applicability	IPC9600		
Example	rv=SetIpcAdiConfA(0)		
Parameters	Parameter Name	Range	Default
	fnId	0	0

5.4.13.3 SetIpcAdiConfB

Description	Set configuration B.
	`STATUS SetIpcAdiConfB(`
	`  IN int32 fnId);`		

Applicability	IPC9600		
Example	rv=SetIpcAdiConfB(0)		
Parameters	Parameter Name	Range	Default
	fnId	0	0

5.4.13.4 SetIpcAdiConfC

Description	Set configuration C.
	`STATUS SetIpcAdiConfC(`
	`  IN int32 fnId);`		

Applicability	IPC9600		
Example	rv=SetIpcAdiConfC(0)		
Parameters	Parameter Name	Range	Default
	fnId	0	0

5.4.13.5 SetIpcAdiConfD

Description	Set configuration D.
	`STATUS SetIpcAdiConfD(`
	`  IN int32 fnId);`		

Applicability	IPC9600		
Example	rv=SetIpcAdiConfD(0)		
Parameters	Parameter Name	Range	Default
	fnId	0	0

5.4.13.6 SetIpcAdiConfE

Description	Set configuration E.
	`STATUS SetIpcAdiConfE(`
	`  IN int32 fnId);`		

Applicability	IPC9600		
Example	rv=SetIpcAdiConfE(0)		
Parameters	Parameter Name	Range	Default

	fnId	0	0

5.4.13.7 SetIpcAdiConfF

Description	Set configuration F.
	`STATUS SetIpcAdiConfF(`
	`  IN int32 fnId);`		

Applicability	IPC9600		
Example	rv=SetIpcAdiConfF(0)		
Parameters	Parameter Name	Range	Default
	fnId	0	0

6 Common Commands Reports, BISTs, Tables and Datasets

6.1 Power Up & Start Operation Reports

Applicability: IPC9xxx, IPC17xx, IPC1603
Commands: SetIpcPtpAdminStart

The following are examples for Power up & Basic operation.

6.1.1 BC

```
00:00:00:00 [#0113] Application load - PASSED
00:00:00:00 [#0503] 1.0.1.0.1.2.2.172
00:00:00:00 [#0113] IPC9000-20-v1.50
00:00:00:00 [#0504] CDB4
00:00:00:00 [#0113] Applying default parameters
 ================================================
 Slave Port BIST Report - Initialization
 ------------------------------------------------
 FPGA read                                - PASSED
 FPGA write & read 1st                    - PASSED
 FPGA write & read 2nd                    - PASSED
 PLL-1                                    - PASSED
 PLL-2                                    - PASSED
 PLL-3                                    - PASSED
 PLL-4                                    - PASSED
 Clock sync init                          - PASSED
 ------------------------------------------------
 Slave Port BIST Initialization           - PASSED
 ================================================
 ================================================
 Master Port BIST Report - Initialization
 ------------------------------------------------
 FPGA read                                - PASSED
 FPGA write & read 1st                    - PASSED
 FPGA write & read 2nd                    - PASSED
 PLL-1                                    - N/A
 PLL-2                                    - N/A
 PLL-3                                    - N/A
 PLL-4                                    - N/A
 Clock sync init                          - PASSED
 ------------------------------------------------
 Master Port BIST Initialization          - PASSED
 ================================================
00:00:00:00 [#1100] [GetIpcAdminIfcNetMacAddr]: 00:0A:35:01:F4:16
00:00:00:00 [#1100] [GetIpcAdminIfcNetIpAddr]: 192.168.1.22
00:00:00:00 [#1100] [GetIpcAdminIfcNetIpSubnetMask]: 255.255.255.0
00:00:00:00 [#1100] [GetIpcAdminIfcNetIpDefaultGateway]: 192.168.1.1
00:00:00:00 [#1100] [GetIpcPtpDsClockIdentity]: 00:0A:35:FF:FE:01:F4:16
00:00:00:00 [#1100] [GetIpcPtpAdminRole]: 4
00:00:00:00 [#0113] IPC9000 Boundary Clock is ready
```

6.1.2 Slave

```
00:00:00:00 [#0113] Application load - PASSED
00:00:00:00 [#0503] 1.0.1.0.1.2.2.172
```

```
00:00:00:00 [#0113] IPC9000-20-v1.50
00:00:00:00 [#0504] CDB4
00:00:00:00 [#0113] Applying default parameters
 ================================================
 Slave Port BIST Report - Initialization
 ------------------------------------------------
 FPGA read                              - PASSED
 FPGA write & read 1st                  - PASSED
 FPGA write & read 2nd                  - PASSED
 PLL-1                                  - PASSED
 PLL-2                                  - PASSED
 PLL-3                                  - PASSED
 PLL-4                                  - PASSED
 Clock sync init                        - PASSED
 ------------------------------------------------
 Slave Port BIST Initialization         - PASSED
 ================================================
00:00:00:00 [#1100] [GetIpcAdminIfcNetMacAddr]: 00:0A:35:01:F4:14
00:00:00:00 [#1100] [GetIpcAdminIfcNetIpAddr]: 192.168.1.20
00:00:00:00 [#1100] [GetIpcAdminIfcNetIpSubnetMask]: 255.255.255.0
00:00:00:00 [#1100] [GetIpcAdminIfcNetIpDefaultGateway]: 192.168.1.1
00:00:00:00 [#1100] [GetIpcPtpDsClockIdentity]: 00:0A:35:FF:FE:01:F4:16
00:00:00:00 [#1100] [GetIpcPtpAdminRole]: 1
00:00:00:00 [#0113] IPC9000 Unicast Slave is ready
```

6.1.3 Master

```
00:00:00:00 [#0113] Application load - PASSED
00:00:00:00 [#0503] 1.0.1.0.1.2.2.172
00:00:00:00 [#0113] IPC9000-20-v1.50
00:00:00:00 [#0504] CDB4
00:00:00:00 [#0113] Applying default parameters
 ================================================
 Master BIST Report - Initialization
 ------------------------------------------------
 FPGA read                              - PASSED
 FPGA write & read 1st                  - PASSED
 FPGA write & read 2nd                  - PASSED
 PLL-1                                  - PASSED
 PLL-2                                  - PASSED
 PLL-3                                  - PASSED
 PLL-4                                  - PASSED
 Clock sync init                        - PASSED
 ------------------------------------------------
 Master Port BIST Initialization        - PASSED
 ================================================
00:00:00:00 [#1100] [GetIpcAdminIfcNetMacAddr]: 00:0A:35:01:F4:15
00:00:00:00 [#1100] [GetIpcAdminIfcNetIpAddr]: 192.168.1.21
00:00:00:00 [#1100] [GetIpcAdminIfcNetIpSubnetMask]: 255.255.255.0
00:00:00:00 [#1100] [GetIpcAdminIfcNetIpDefaultGateway]: 192.168.1.1
00:00:00:00 [#1100] [GetIpcPtpDsClockIdentity]: 00:0A:35:FF:FE:01:F4:15
00:00:00:00 [#1100] [GetIpcPtpAdminRole]: 0
00:00:00:00 [#0113] IPC9000 Unicast Master is ready
```

6.2 Power Up & Start Operation Reports – G.8275.1

Applicability: IPC9xxx, IPC17xx
Commands: SetIpcPtpAdminStart

The following are examples for Power up & Basic operation.

6.2.1 BC

```
Application load - PASSED
2.0.1.0.1.2.2.205
IPC9600-20-1.00
CDB4
Applying default parameters
================================================
Slave Port BIST Report - Initialization
```

```
------------------------------------------------
FPGA read                                - PASSED
FPGA write & read 1st                    - PASSED
FPGA write & read 2nd                    - PASSED
PLL-1                                    - PASSED
PLL-2                                    - PASSED
PLL-3                                    - PASSED
PLL-4                                    - PASSED
Clock sync init                          - PASSED
------------------------------------------------
Slave Port BIST Initialization           - PASSED
================================================
================================================
Master Port BIST Report - Initialization
------------------------------------------------
FPGA read                                - PASSED
FPGA write & read 1st                    - PASSED
FPGA write & read 2nd                    - PASSED
PLL-1                                    - N/A
PLL-2                                    - N/A
PLL-3                                    - N/A
PLL-4                                    - N/A
Clock sync init                          - PASSED
------------------------------------------------
Master Port BIST Initialization          - PASSED
================================================
00:00:00:01 [#1100] [GetIpcAdminIfcNetMacAddr]: 00:0A:35:01:F4:16
00:00:00:01 [#1100] [GetIpcPtpDsClockIdentity]: 00:0A:35:FF:FE:01:F4:16
00:00:00:01 [#1100] [GetIpcPtpAdminRole]: 24
00:00:00:01 [#0113] IPC9600 T-BC is ready
```

6.2.2 Slave

```
Application load - PASSED
2.0.1.0.1.2.2.205
IPC9600-20-1.00
CDB4
Applying default parameters
================================================
Slave Port BIST Report - Initialization
------------------------------------------------
FPGA read                                - PASSED
FPGA write & read 1st                    - PASSED
FPGA write & read 2nd                    - PASSED
PLL-1                                    - PASSED
PLL-2                                    - PASSED
PLL-3                                    - PASSED
Clock sync init                          - PASSED
------------------------------------------------
Slave Port BIST Initialization           - PASSED
================================================
00:00:00:01 [#1100] [GetIpcAdminIfcNetMacAddr]: 00:0A:35:01:F4:14
00:00:00:01 [#1100] [GetIpcPtpDsClockIdentity]: 00:0A:35:FF:FE:01:F4:14
00:00:00:01 [#1100] [GetIpcPtpAdminRole]: 23
00:00:00:01 [#0113] IPC9600 T-TSC is ready
```

6.2.3 Master

```
Application load - PASSED
2.0.1.0.1.2.2.205
IPC9600-20-1.00
CDB4
Applying default parameters
================================================
Master Port BIST Report - Initialization
------------------------------------------------
FPGA read                                - PASSED
FPGA write & read 1st                    - PASSED
FPGA write & read 2nd                    - PASSED
PLL-1                                    - PASSED
```

```
PLL-2                                       - PASSED
PLL-3                                       - PASSED
PLL-4                                       - PASSED
Clock sync init                             - PASSED
------------------------------------------------
Master Port BIST Initialization             - PASSED
================================================
00:00:00:01 [#1100] [GetIpcAdminIfcNetMacAddr]: 00:0A:35:01:F4:15
00:00:00:01 [#1100] [GetIpcPtpDsClockIdentity]: 00:0A:35:FF:FE:01:F4:15
00:00:00:01 [#1100] [GetIpcPtpAdminRole]: 22
00:00:00:00 [#0113] IPC9600 T-GM: Master is ready
```

6.3 Power Up & Start Operation Reports – IPC9600 & External Device

Applicability: IPC9600, extDeviceMode=4x
Commands: SetIpcPtpAdminStart

The following are examples for Power up & Basic operation.

6.3.1 BC

```
00:00:00:00 [#0113] Application load - PASSED
00:00:00:00 [#0503] 1.0.1.0.1.2.2.172
00:00:00:00 [#0113] IPC9600-20-v1.00
00:00:00:00 [#0504] CDB4
00:00:00:00 [#0113] Applying default parameters
 ================================================
 Slave Port BIST Report - Initialization
 ------------------------------------------------
 FPGA read                                  - PASSED
 FPGA write & read 1st                      - PASSED
 FPGA write & read 2nd                      - PASSED
 PLL-1                                      - PASSED
 PLL-2                                      - PASSED
 PLL-3                                      - PASSED
 PLL-4                                      - PASSED
 PLL-5                                      - PASSED
 PLL-5N                                     - PASSED
 ED detect                                  - PASSED
 ED init                                    - PASSED
 Clock sync init                            - PASSED
 ------------------------------------------------
 Slave Port BIST Initialization             - PASSED
 ================================================
 ================================================
 Master Port BIST Report - Initialization
 ------------------------------------------------
 FPGA read                                  - PASSED
 FPGA write & read 1st                      - PASSED
 FPGA write & read 2nd                      - PASSED
 PLL-1                                      - N/A
 PLL-2                                      - N/A
 PLL-3                                      - N/A
 PLL-4                                      - N/A
 PLL-5                                      - N/A
 PLL-5N                                     - N/A
 Clock sync init                            - PASSED
 ------------------------------------------------
 Master Port BIST Initialization            - PASSED
 ================================================
00:00:00:00 [#1100] [GetIpcAdminIfcNetMacAddr]: 00:0A:35:01:F4:16
00:00:00:00 [#1100] [GetIpcAdminIfcNetIpAddr]: 192.168.1.22
00:00:00:00 [#1100] [GetIpcAdminIfcNetIpSubnetMask]: 255.255.255.0
00:00:00:00 [#1100] [GetIpcAdminIfcNetIpDefaultGateway]: 192.168.1.1
00:00:00:00 [#1100] [GetIpcPtpDsClockIdentity]: 00:0A:35:FF:FE:01:F4:16
00:00:00:00 [#1100] [GetIpcPtpAdminRole]: 4
00:00:00:00 [#1100] [GetIpcAdminExtDeviceMode]:41
00:00:00:00 [#0113] IPC9600 Boundary Clock is ready
```

6.3.2 Slave

```
00:00:00:00 [#0113] Application load - PASSED
00:00:00:00 [#0503] 1.0.1.0.1.2.2.172
00:00:00:00 [#0113] IPC9600-20-v1.00
00:00:00:00 [#0504] CDB4
00:00:00:00 [#0113] Applying default parameters
 ================================================
 Slave Port BIST Report - Initialization
 ------------------------------------------------
 FPGA read                                - PASSED
 FPGA write & read 1st                    - PASSED
 FPGA write & read 2nd                    - PASSED
 PLL-1                                    - PASSED
 PLL-2                                    - PASSED
 PLL-3                                    - PASSED
 PLL-4                                    - PASSED
 PLL-5                                    - PASSED
 PLL-5N                                   - PASSED
 ED detect                                - PASSED
 ED init                                  - PASSED
 Clock sync init                          - PASSED
 ------------------------------------------------
 Slave Port BIST Initialization           - PASSED
 ================================================
00:00:00:00 [#1100] [GetIpcAdminIfcNetMacAddr]: 00:0A:35:01:F4:14
00:00:00:00 [#1100] [GetIpcAdminIfcNetIpAddr]: 192.168.1.20
00:00:00:00 [#1100] [GetIpcAdminIfcNetIpSubnetMask]: 255.255.255.0
00:00:00:00 [#1100] [GetIpcAdminIfcNetIpDefaultGateway]: 192.168.1.1
00:00:00:00 [#1100] [GetIpcPtpDsClockIdentity]: 00:0A:35:FF:FE:01:F4:14
00:00:00:00 [#1100] [GetIpcPtpAdminRole]: 1
00:00:00:00 [#1100] [GetIpcAdminExtDeviceMode]:41
00:00:00:00 [#0113] IPC9600 Unicast Slave is ready
```

6.3.3 Master

```
00:00:00:00 [#0113] Application load - PASSED
00:00:00:00 [#0503] 1.0.1.0.1.2.2.172
00:00:00:00 [#0113] IPC9600-20-v1.00
00:00:00:00 [#0504] CDB4
00:00:00:00 [#0113] Applying default parameters
 ================================================
 Master Port BIST Report - Initialization
 ------------------------------------------------
 FPGA read                                - PASSED
 FPGA write & read 1st                    - PASSED
 FPGA write & read 2nd                    - PASSED
 PLL-1                                    - PASSED
 PLL-2                                    - PASSED
 PLL-3                                    - PASSED
 PLL-4                                    - PASSED
 PLL-5                                    - PASSED
 PLL-5N                                   - PASSED
 Clock sync init                          - PASSED
 ------------------------------------------------
 Master Port BIST Initialization          - PASSED
 ================================================
00:00:00:00 [#1100] [GetIpcAdminIfcNetMacAddr]: 00:0A:35:01:F4:15
00:00:00:00 [#1100] [GetIpcAdminIfcNetIpAddr]: 192.168.1.21
00:00:00:00 [#1100] [GetIpcAdminIfcNetIpSubnetMask]: 255.255.255.0
00:00:00:00 [#1100] [GetIpcAdminIfcNetIpDefaultGateway]: 192.168.1.1
00:00:00:00 [#1100] [GetIpcPtpDsClockIdentity]: 00:0A:35:FF:FE:01:F4:15
00:00:00:00 [#1100] [GetIpcPtpAdminRole]: 0
00:00:00:00 [#1100] [GetIpcAdminExtDeviceMode]:44
00:00:00:00 [#0113] IPC9600 Unicast Master is ready
```

6.4 Built In Self Test (BIST)

Applicability: IPC9xxx, IPC17xx, IPC1603
Commands: GetIpcAdminBistRepMasterInit

GetIpcAdminBistRepMasterShowtime
GetIpcAdminBistRepSlaveInit
GetIpcAdminBistRepSlaveShowtime
GetIpcAdminBistRepNativePll
GetIpcAdminBistRepPll

In BC mode, use Master and Slave BIST reports.

6.4.1 Initialization

6.4.1.1 Slave & BC

The following is an example for the BIST report Initialization.

```
================================================
 Slave Port BIST Report - Initialization
 ------------------------------------------------
 FPGA read                              - PASSED
 FPGA write & read 1st                  - PASSED
 FPGA write & read 2nd                  - PASSED
 PLL-1                                  - PASSED
 PLL-2                                  - PASSED
 PLL-3                                  - PASSED
 PLL-4                                  - PASSED
 Clock sync init                        - PASSED
 ------------------------------------------------
 Slave Port BIST Initialization         - PASSED
 ================================================
```

The following is an example for the BIST report Initialization for ExtDevMode=4x & IPC9600.

```
================================================
 Slave Port BIST Report - Initialization
 ------------------------------------------------
 FPGA read                              - PASSED
 FPGA write & read 1st                  - PASSED
 FPGA write & read 2nd                  - PASSED
 PLL-1                                  - PASSED
 PLL-2                                  - PASSED
 PLL-3                                  - PASSED
 PLL-4                                  - PASSED
 PLL-5                                  - PASSED
 PLL-5N                                 - PASSED
 ED detect                              - PASSED
 ED init                                - PASSED
 Clock sync init                        - PASSED
 ------------------------------------------------
 Slave Port BIST Initialization         - PASSED
 ================================================
```

The report shows BIST result during initialization and includes the following:

Parameter	Description
FPGA read	Reading known data from a pre-determined register
FPGA write & read 1st	Writing a data to one of the registers and reading back
FPGA write & read 2nd	Writing a data to one of the registers and reading back
PLL-1	PLL-1 is locked (SYS PLL)
PLL-2	PLL-2 is locked (TOD Freq PLL)
PLL-3	PLL-3 is locked (TOD DPLL)
PLL-4	PLL-4 is locked (CLKOUT PLL)
PLL-5	PLL-5 is locked (SYSIN/A PLL)
PLL-5N	PLL-5 is in normal operation (SYSIN/A PLL)
ED detect	External device detection
ED init	External device initialization
Clock sync init	Initialization of clock synchronization algorithm
BIST Initialization	Overall BIST initialization result

All the PLL BIST parametres reports information related to device conditions during initialization stage.

In case of extDev=45,47 ED detect PASS only in case the two external devices were detected..

6.4.1.2 Master & BC

The following is an example for the BIST report Initialization.

```
================================================
Master Port BIST Report - Initialization
------------------------------------------------
FPGA read                               - PASSED
FPGA write & read 1st                   - PASSED
FPGA write & read 2nd                   - PASSED
PLL-1                                   - PASSED
PLL-2                                   - PASSED
PLL-3                                   - PASSED
PLL-4                                   - PASSED
Clock sync init                         - PASSED
------------------------------------------------
Master Port BIST Initialization         - PASSED
================================================
```

The following is an example for the BIST report Initialization for IPC9600.

```
================================================
Master Port BIST Report - Initialization
------------------------------------------------
FPGA read                               - PASSED
FPGA write & read 1st                   - PASSED
FPGA write & read 2nd                   - PASSED
PLL-1                                   - PASSED
PLL-2                                   - PASSED
PLL-3                                   - PASSED
PLL-4                                   - PASSED
PLL-5                                   - PASSED
PLL-5N                                  - PASSED
Clock sync init                         - PASSED
------------------------------------------------
Master Port BIST Initialization         - PASSED
================================================
```

The report shows BIST result during initialization and includes the following:

Parameter	Description
FPGA read	Reading known data from a pre-determined register
FPGA write & read 1st	Writing a data to one of the registers and reading back
FPGA write & read 2nd	Writing a data to one of the registers and reading back
PLL-1	PLL-1 is locked (SYS PLL)
PLL-2	PLL-2 is locked (TOD Freq PLL)
PLL-3	PLL-3 is locked (TOD DPLL)
PLL-4	PLL-4 is locked (CLKOUT PLL)
PLL-5	PLL-5 is locked (SYSIN/A PLL)
PLL-5N	PLL-5 is in normal operation (SYSIN/A PLL)
Clock sync init	Clock in synchronization init
BIST Initialization	Overall BIST initialization result

All the PLL BIST parametres reports information related to device conditions during initialization stage.

6.4.2 Showtime

6.4.2.1 Slave & BC

The following is an example for the BIST report Showtime.

```
================================================
Slave Port BIST Report - Showtime
```

```
-----------------------------------------------
Local clock                          - PASSED
Receive Announce                     - PASSED
Communication Path                   - PASSED
State                                - PASSED
Frequency / Osc offset               - PASSED
PDV                                  - PASSED
Selected master lock                 - PASSED
Rx classifier                        - PASSED
-----------------------------------------------
Slave Port BIST Showtime             - PASSED
===============================================
```

The following is an example for the BIST report Showtime for IPC9600.

```
===============================================
Slave Port BIST Report - Showtime
-----------------------------------------------
Local clock                          - PASSED
Receive Announce                     - PASSED
Communication Path                   - PASSED
State                                - PASSED
Frequency / Osc offset               - PASSED
PDVi M2S                             - PASSED
FPP M2S                              - PASSED
FPP S2M                              - PASSED
PDV M2S                              - PASSED
PDV S2M                              - PASSED
Selected master lock                 - PASSED
Rx classifier                        - PASSED
ED clock valid                       - PASSED
ED control                           - PASSED
ED track                             - PASSED
Global State                         - PASSED
-----------------------------------------------
Slave Port BIST Showtime             - PASSED
===============================================
```

The report shows BIST during Showtime when in Slave mode and includes the following:

Parameter	Description
Local clock	Local oscillator detected
Receive Announce	Valid Announce packets received
Communication Path	Communication path status
State	State is lock
Frequency / Osc offset	Local oscillator frequency offset is within limits (<=5ppm)
PDVi	PDV is within limits (<=10ms)
FPP M2S	FPP% x 10 M2S is within limits (=>1%)
FPP S2M	FPP% x 10 S2M is within limits (=>1%)
PDV M2S	PDV M2S is within limits (<=10ms)
PDV S2M	PDV S2M is within limits (<=10ms)
Selected master lock	Selected (Parent) Master is reporting to be in Lock state
Rx classifier	Hardware classifier is receiving and detecting 1588 v2 packets
ED clock valid	External device 1PPS clock connected to CLKIN is valid (applicable for extDevMode=41,45)
ED control	External device is being controlled
ED Track	Exetrnal device 1PPS is tracked within limites
Global State	Global state is lock
BIST Showtime	Overall BIST Showtime result

PDVi and PDV are based on two separate PDV estimators.
PDVi applicable only for M2S (Master to Slave)
PDVi has better immunity than PDV to the servo internal frequency and phase recovery errors.

6.4.2.2 Master & BC

The following is an example for the BIST report Showtime.

```
==================================
Master Port BIST Report - Showtime
----------------------------------
Interrupts                - PASSED
Local clock               - PASSED
1PPS Lock                 - PASSED
Timestamp                 - PASSED
Ch TS  SKA C
00 P   P   P
01 P   P   P
02 P   P   P
03 P   P   P
04 P   P   P
05 P   P   P
06 P   P   P
07 P   P   P
08 P   P   P
09 P   P   P
10 P   P   P
----------------------------------
Master Port BIST Showtime - PASSED
==================================
```

The report shows BIST during Showtime when in Master mode and includes the following:

Parameter	Description
Interrupts	Local interrupts
Local clock	Local oscillator detected
1PPS Lock	Master is locked to 1PPS signal
Timestamp	Hardware timestamp is active
Ch	Channel number
SKA	Slave Keep Alive – master keep on receiving either Delay_Req or RUTS (REQUEST UNICAST TRANSMISSION TLV) from slave
C	Communication path
BIST Showtime	Overall BIST Showtime result

6.4.3 Others

6.4.3.1 GetIpcAdminBistRepNativePll

The following is an example for the BIST Native PLL for IPC9000.

```
================================================
BIST Report - Native PLL
------------------------------------------------
PLL-1 native lock                    - PASSED
PLL-2 native lock                    - PASSED
PLL-3 native lock                    - PASSED
PLL-4 native lock                    - PASSED
PLL-4 native normal                  - PASSED
------------------------------------------------
BIST Native PLL                      - PASSED
================================================
```

The following is an example for the BIST Native PLL for IPC9600

```
================================================
BIST Report - Native PLL
------------------------------------------------
PLL-1 native lock                    - PASSED
PLL-2 native lock                    - PASSED
PLL-3 native lock                    - PASSED
PLL-4 native lock                    - PASSED
PLL-4 native normal                  - PASSED
PLL-5 native lock                    - PASSED
PLL-5 native normal                  - PASSED
------------------------------------------------
```

```
BIST Native PLL                          - PASSED
================================================
```

6.4.3.2 GetIpcAdminBistRepPll

The following is an example for the BIST PLL for IPC9000.

```
================================================
BIST Report - PLL
------------------------------------------------
PLL-1 lock                               - PASSED
PLL-2 lock                               - PASSED
PLL-3 lock                               - PASSED
PLL-4 lock                               - PASSED
PLL-4 normal                             - PASSED
------------------------------------------------
BIST PLL                                 - PASSED
================================================
```

The following is an example for the BIST PLL for IPC9600.

```
================================================
BIST Report - PLL
------------------------------------------------
PLL-1 lock                               - PASSED
PLL-2 lock                               - PASSED
PLL-3 lock                               - PASSED
PLL-4 lock                               - PASSED
PLL-4 normal                             - PASSED
PLL-5 lock                               - PASSED
PLL-5 normal                             - PASSED
------------------------------------------------
BIST PLL                                 - PASSED
================================================
```

The report shows the current PLL BIST includes the following:

Parameter	Description
PLL-1	PLL-1 is locked (SYS PLL)
PLL-2	PLL-2 is locked (TOD Freq PLL)
PLL-3	PLL-3 is locked (TOD DPLL)
PLL-4	PLL-4 is locked (CLKOUT PLL)
PLL-4N	PLL-4 is in normal operation (CLKOUT PLL)
PLL-5	PLL-5 is locked (SYSIN/A PLL)
PLL-5N	PLL-5 is in normal operation (SYSIN/A PLL)
BIST PLL	Overall BIST PLL result

6.5 GetIpcAdminIfcNetConfig

Applicability: IPC9xxx, IPC17xx, IPC1603
Commands: GetIpcAdminIfcNetConfig

The following is an example for the network interface configuration.

```
==================================
Network Interface Configuration
----------------------------------
Number              0
Enable              1
Speed               2
MAC Addr            FC:C2:40:00:00:A6
IP Addr             192.168.111.121
Subnet Mask         255.255.255.000
Default Gateway     192.168.111.1
==================================
```

The network interface configuration includes the following parameters:

Parameter	Description
Number	Interface number. 0 only

Enable	Enable/Disable 0 – Disable, 1 – Enable
Speed	MAC Speed: 0 – Unknown, 1 – 100Mbps, 2 – 1000Mbps
MAC Addr	MAC address
IP Addr	IP address
Subnet Mask	Subnet Mask
Default Gateway	Default Gateway

6.6 GetIpcPtpState

Applicability: IPC9xxx, IPC17xx, IPC1603
Commands: GetIpcPtpStateSlaveAll
GetIpcPtpStateMasterAll

The following are examples for the state report.

```
================================
Slave State
--------------------------------
State                     = 4
State Extended            = 8
FreeRun  Elapse Time      = 0
Trace    Elapse Time      = 0
Lock     Elapse Time      = 7028
Holdover Elapse Time      = 0
FreeRun  Time (sec.)      = 1
Trace    Time (sec.)      = 32
Lock     Time (sec.)      = 1167
Holdover Time (sec.)      = 0
================================

================================
Master State
--------------------------------
State                     = 3
FreeRun  Elapse Time      = 0
Trace    Elapse Time      = 0
Lock     Elapse Time      = 19
Holdover Elapse Time      = 0
FreeRun  Time (sec.)      = 44
Trace    Time (sec.)      = 58
Lock     Time (sec.)      = 243
Holdover Time (sec.)      = 0
================================
```

The elapsed time is shown only for the current state.

6.7 GetIpcAdminIfcNetPktRep

Applicability: IPC9xxx, IPC17xx, IPC1603
Commands: SetIpcAdminIfcNetPerPktRepEn
GetIpcAdminIfcNetPktRep1Min
GetIpcAdminIfcNetPktRep15Min
GetIpcAdminIfcNetPktRep60Min
GetIpcAdminIfcNetPktRepInf

The following are examples for the packet report.

```
====================================
Slave Port Packet Report - 60 minutes
------------------------------------
Uni Pkt Rx                = 113546
Uni Min [pkt/sec]         = 31
Uni Max [pkt/sec]         = 33
Uni Avg [pkt/sec]         = 32
Multi Pkt Rx              = 0
Multi Min [pkt/sec]       = 0
Multi Max [pkt/sec]       = 0
```

```
Multi Avg [pkt/sec]       = 0
====================================

====================================
Master Port Packet Report - 60 minutes
------------------------------------
Uni Pkt Rx                = 864998
Uni Min [pkt/sec]         = 182
Uni Max [pkt/sec]         = 430
Uni Avg [pkt/sec]         = 244
Multi Pkt Rx              = 0
Multi Min [pkt/sec]       = 0
Multi Max [pkt/sec]       = 0
Multi Avg [pkt/sec]       = 0
====================================
```

The report includes the following network packet statistics:

Parameter	Description
Uni Pkt Rx	Number of unicast packets received
Uni Min [pkt/sec]	Minimum unicast packet/sec received
Uni Max [pkt/sec]	Maximum unicast packet/sec received
Uni Avg [pkt/sec]	Average unicast packet rate in packet/sec received
Multi Pkt Rx	Number of multicast packets received
Multi Min [pkt/sec]	Minimum multicast packet/sec received
Multi Max [pkt/sec]	Maximum multicast packet/sec received
Multi Avg [pkt/sec]	Average multicast packet/sec received

6.8 GetIpcPtpCntRep

Applicability: IPC9xxx, IPC17xx, IPC1603
Command: GetIpcPtpCntRepSlave
GetIpcPtpCntRepMaster

The following are examples for the PTP slave port packet counter report.

```
================================
Slave Port Packet Counter Report
--------------------------------
Rx An             = 205
Rx An Drop        = 0
Rx Sync           = 16902
Rx Sync Miss      = 0
Rx Sync MissOrd   = 0
Rx Sync Drop      = 0
Rx FU             = 16902
Rx FU Miss        = 0
Rx FU Drop        = 0
Rx Dresp          = 16903
Rx Dresp Miss     = 0
Rx Dresp MissOrd  = 0
Rx Dresp Drop     = 0
Rx Dresp RspDrop  = 0
Tx Dreq           = 16902
================================
```

The report includes the following network packet statistics:

Parameter	Description
Rx An	Number of received Announce packets
Rx An Drop	Number of dropped Announce packets
Rx Sync	Number of received Sync packets
Rx Sync Miss	Number of missing Sync packets
Rx Sync MissOrd	Number of missordered Sync packets
Rx Sync Drop	Number of dropped Sync packets
Rx FU	Number of received FU packets
Rx FU Miss	Number of missing FU packets

Rx FU Drop	Number of dropped FU packets
Rx Dresp	Number of received Dresp packets
Rx Dresp Miss	Number of missing Dresp packets
Rx Dresp MissOrd	Number of missordered Dresp packets
Rx Dresp Drop	Number of dropped Dresp packets
Rx Dresp RspDrop	Number of Requesting Source Port identity dropped Dresp packets
Tx Dreq	Number of transmitted Dreq packets

The following are examples for the PTP master port packet counter report.

```
==================================
Master Port Packet Counter Report
----------------------------------
Tx An           = 206
Tx Sync         = 12889
Tx FU           = 0
Rx Dreq         = 3221
Rx Dreq Drop1   = 0
Rx Dreq Drop2   = 0
Rx Dreq Drop3   = 0
Tx Dresp        = 3221
==================================
```

The report includes the following network packet statistics:

Parameter	Description
Tx An	Number of transmitted Announce packets
Tx Sync	Number of transmitted Sync packets
Tx FU	Number of transmitted FU packets
Rx Dreq	Number of received Dreq packets
Rx Dreq Drop1	Number of dropped Dreq packets due to Dreq overrun (by the next Dreq)
Rx Dreq Drop2	Number of dropped Dreq packets due to other reasons
Rx Dreq Drop3	Number of dropped Dreq packets due to non-allocated channel
Tx Dresp	Number of transmitted Dresp packets

6.9 GetIpcPtpCntRepSignaling

Applicability: IPC9xxx, IPC17xx
Command: GetIpcPtpCntRepSignaling

The following are examples for the PTP signaling packet counter report.

```
==========================================
      Signaling Packet Counter Report
------------------------------------------
                    Ann      Sync   DelResp
------------------------------------------
TxSigRuts             0         0         0
RxSigRuts           958       958       958
TxSigGruts          958       958       958
RxSigGruts            0         0         0
TxSigDgruts           0         0         0
RxSigDgruts           0         0         0
TxSigCuts             0         0         0
RxSigCuts             0         0         0
TxSigAcuts            0         0         0
RxSigAcuts            0         0         0
==========================================
```

The report includes the following network packet statistics:

Parameter	Description
TxSigRuts	Number of transmitted REQUEST_UNICAST_TRANSMISSION packets

RxSigRuts	Number of received REQUEST_UNICAST_TRANSMISSION packets
TxSigGruts	Number of transmitted GRANT_UNICAST_TRANSMISSION packets
RxSigGruts	Number of received GRANT_UNICAST_TRANSMISSION packets
TxSigDgruts	Number of transmitted GRANT_UNICAST_TRANSMISSION + durationField=0 packets
RxSigDgruts	Number of received GRANT_UNICAST_TRANSMISSION + durationField=0 packets
TxSigCuts	Number of transmitted CANCEL_UNICAST_TRANSMISSION packets
RxSigCuts	Number of received CANCEL_UNICAST_TRANSMISSION packets
TxSigAcuts	Number of transmitted ACKNOWLEDGE_CANCEL_UNICAST_TRANSMISSION packets
RxSigAcuts	Number of received ACKNOWLEDGE_CANCEL_UNICAST_TRANSMISSION packets

6.10 GetIpcPtpCntRepSignalingLastMin

Applicability: IPC9xxx, IPC17xx
Command: GetIpcPtpCntRepSignalingLastMin

The following are examples for the PTP signaling packet Last Minute counter report.

```
==========================================
Signaling Last Minute Packet Counter Report
------------------------------------------
                    Ann      Sync   DelResp
------------------------------------------
TxSigRuts             0         0         0
RxSigRuts            10        10        10
TxSigGruts           10        10        10
RxSigGruts            0         0         0
TxSigDgruts           0         0         0
RxSigDgruts           0         0         0
TxSigCuts             0         0         0
RxSigCuts             0         0         0
TxSigAcuts            0         0         0
RxSigAcuts            0         0         0
==========================================
```

The report includes the following network packet statistics:

Parameter	Description
TxSigRuts	Number of transmitted REQUEST_UNICAST_TRANSMISSION packets during last minute
RxSigRuts	Number of received REQUEST_UNICAST_TRANSMISSION packets during last minute
TxSigGruts	Number of transmitted GRANT_UNICAST_TRANSMISSION packets during last minute
RxSigGruts	Number of received GRANT_UNICAST_TRANSMISSION packets during last minute
TxSigDgruts	Number of transmitted GRANT_UNICAST_TRANSMISSION + durationField=0 packets during last minute
RxSigDgruts	Number of received GRANT_UNICAST_TRANSMISSION + durationField=0 packets during last minute
TxSigCuts	Number of transmitted CANCEL_UNICAST_TRANSMISSION packets during last minute
RxSigCuts	Number of received CANCEL_UNICAST_TRANSMISSION packets during last minute
TxSigAcuts	Number of transmitted ACKNOWLEDGE_CANCEL_UNICAST_TRANSMISSION packets during last minute
RxSigAcuts	Number of received ACKNOWLEDGE_CANCEL_UNICAST_TRANSMISSION packets during last minute

6.11 Network Performance Monitoring (NPM)

6.11.1 GetIpcPtpNpmNetworkRep

Applicability: IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Commands: GetIpcPtpNpmNetworkRep1min

The following is an example for the network report.

```
=====================================
Slave Port Network Report - Last minute
-------------------------------------
Slave Status              = LK
Communication Path        = PASS
Elapsed Time [sec]        = 6987
Frequency Offset [ppb]    = 513
M2S-PDVi[ns]              = 130
M2S-PDV [ns]              = 130
S2M-PDV [ns]              = 110
M2S-FFP [10x%]            = 1000
S2M-FFP [10x%]            = 1000
RTD [us]                  = 205
RTDU [ns]                 = 50
FFOFF [ppb]               = 4
=====================================
```

Applicability: IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Commands: SetIpcPtpNpmPeriodicNetworkRepEn
GetIpcPtpNpmNetworkRep15min
GetIpcPtpNpmNetworkRep60min
GetIpcPtpNpmNetworkRepInf

The following is an example for the network report.

```
=====================================
Slave Port Network Report - 60 minutes
-------------------------------------
Slave Status              = LK
Communication Path        = PASS
Frequency Offset [ppb]    = 697
Elapsed Time [sec]        = 2716
M2S-PDVi-Min [us]         = 0
M2S-PDVi-Max [us]         = 238 us
M2S-PDVi-Avg [us]         = 96
M2S-PDV-Min  [us]         = 0
M2S-PDV-Max  [us]         = 238
M2S-PDV-Avg  [us]         = 96
S2M-PDV-Min  [us]         = 0
S2M-PDV-Max  [us]         = 142
S2M-PDV-Avg  [us]         = 75
M2S-FPP-Min  [10x%]       = 1000
M2S-FPP-Max  [10x%]       = 1000
M2S-FPP-Avg  [10x%]       = 1000
S2M-FFP-Min  [10x%]       = 1000
S2M-FFP-Max  [10x%]       = 1000
S2M-FFP-Avg  [10x%]       = 1000
RTD-Min   [us]            = 164
RTD-Max   [us]            = 165
RTD-Avg   [us]            = 164
RTDU-Min  [ns]            = 40
RTDU-Max  [ns]            = -140
RTDU-Avg  [ns]            = -26
FFOFF     [ppb]           = 4
=====================================
```

The report includes the following network statistics:

Parameter	Description
Slave Status	LK – Lock, TR – Trace, HO – Holdover, FR – Free Run

Communication Path	Communication path Pass or Fail
Frequency Offset	Frequency Offset between the Master frequency and the Slave local frequency in ppb as provided to OSCIN, SYSIN, depends on the device and the mode of operation.
Elapsed Time	Elapsed time in seconds
M2S-PDVi-Min	Minimum M2S-PDVi measured in ns/us/ms
M2S-PDVi-Max	Maximum M2S-PDVi measured in ns/us/ms
M2S-PDVi-Avg	Average M2S-PDVi measured in ns/us/ms
M2S-PDV-Min	Minimum M2S-PDV measured in ns/us/ms
M2S-PDV-Max	Maximum M2S-PDV measured in ns/us/ms
M2S-PDV-Avg	Average M2S-PDV measured in ns/us/ms
S2M-PDV-Min	Minimum S2M-PDV measured in ns/us/ms
S2M-PDV-Max	Maximum S2M-PDV measured in ns/us/ms
S2M-PDV-Avg	Average S2M-PDV measured in ns/us/ms
M2S-FPP-Min	Minimum M2S-FPP measured in 10x%
M2S-FPP-Max	Maximum M2S-FPP measured in 10x%
M2S-FPP-Avg	Average M2S-FPP measured in 10x%
S2M-FPP-Min	Minimum S2M-FPP measured in 10x%
S2M-FPP-Max	Maximum S2M-FPP measured in 10x%
S2M-FPP-Avg	Average S2M-FPP measured in 10x%
RTD-Min	Minimum round trip delay measured in µsec.
RTD-Max	Maximum round trip delay measured in µsec
RTD-Avg	Average round trip delay measured in µsec
RTDU-Min	Minimum round trip delay uncertainty measured in nsec
RTDU-Max	Maximum round trip delay uncertainty measured in nsec
RTDU-Avg	Average round trip delay uncertainty measured in nsec
FFOFF	Maximal measured Fractional Frequency Offset (FFOFF) in ppb

Note *RTD-Min represents the absolute delay in the network. It defines as the delay of the fastest packets from the master to the slave. In the case of symmetrical network, the RTD-Min/2 would be the delay between the Slave and the Master and is referred as the Path Delay. The difference between RTD-Min and RTD-Max would represent the fluctuations in network delay due to the network behavior.*

Note *The RTDU is providing an estimation of the peak error on RTD measurements. In the example above, RTD-Min = 164µsec while RTDU-Min = 40nsec. Therefore, worst case RTD-Min would be 164µsec + 40nsec = 164.04µsec.*

Note *PDV-Max define the ns/us/ms.*

Note *FFOFF is the maximal measured Fractional Frequency Offset (FFOFF) over sliding window of 5 sec. This is the maximal change of the recovered frequency. The FFOFF provides indication for the frequency stability which is a function of the master frequency stability, the slave frequency stability and the servo clock recovery stability. In case of servo in Lock state with calm conditions, the FFOFF shall be low as several ppbs and down to 0ppb.*

Note *PDVi and PDV are based on two separate PDV estimators.*
PDVi applicable only for M2S (Master to Slave)
PDVi has better immunity than PDV to the servo internal frequency and phase recovery errors.

6.11.2 Packet Delay Variation (PDV) Histogram

Applicability: IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Commands: SetIpcPtpNpmPeriodicPdvHistEn
GetIpcPtpNpmPeriodicPdvHistEn
GetIpcPtpNpmPdvHist
SetIpcPtpNpmPdvHistClr

The following is an example for the PDV histogram report.

```
=====================================
Report: M2S PDV Histogram
Max [ns]     190
```

```
FPC-N/M/V/T  63782 63782 63782 63782
FPP-N/M/V    1000 1000 1000
ET [sec]     4199
---------------------------------------
   BIN      Px10 PKT      N
   0:0.5    1000 63782    12756400
 0.5:5      0    0        0
   5:10     0    0        0
  10:50     0    0        0
  50:100    0    0        0
 100:150    0    0        0
 150:200    0    0        0
 200:300    0    0        0
 300:400    0    0        0
 400:500    0    0        0
 500:600    0    0        0
 600:700    0    0        0
 700:800    0    0        0
 800:900    0    0        0
 900:1m     0    0        0
=======================================
=======================================
Report: S2M PDV Histogram
Max [ns]     170
FPC-N/M/V/T  33600 33600 33600 33600
FPP-N/M/V    1000 1000 1000
ET [sec]     4199
---------------------------------------
   BIN      Px10 PKT      N
   0:0.5    1000 33600    6720000
 0.5:5      0    0        0
   5:10     0    0        0
  10:50     0    0        0
  50:100    0    0        0
 100:150    0    0        0
 150:200    0    0        0
 200:300    0    0        0
 300:400    0    0        0
 400:500    0    0        0
 500:600    0    0        0
 600:700    0    0        0
 700:800    0    0        0
 800:900    0    0        0
 900:1m     0    0        0
=======================================
```

The periodic PDV histogram is comprised of two main parts: Master to Slave (M2S) PDV histogram and Slave to Master (S2M) PDV Histogram. The M2S PDV histogram is based on timestamps t_1 and t_2 derived from the received Sync packets whereas the S2M PDV histogram is based on t_3 and t_4 derived from the transmitted Delay Request and received Delay Response.
The M2S PDV Histogram starts with a header shown below.

```
=======================================
Report: M2S PDV Histogram
Max [ns]     190
FPC-N/M/V/T  63782 63782 63782 63782
FPP-N/M/V    1000 1000 1000
ET [sec]     4199
---------------------------------------
```

The M2S PDV Histogram continues with the PDV histogram shown below.

```
 BIN       Px10 PKT      N
   0:0.5   1000 33600    6720000
 0.5:5     0    0        0
   5:10    0    0        0
  10:50    0    0        0
  50:100   0    0        0
 100:150   0    0        0
 150:200   0    0        0
```

```
200:300    0     0         0
300:400    0     0         0
400:500    0     0         0
500:600    0     0         0
600:700    0     0         0
700:800    0     0         0
800:900    0     0         0
900:1m     0     0         0
```

The PDV is divided into non-uniform bins shown in µsec. The first bin is up to 0.5µsec, the second is 0.5 µsec to 5µsec, the third is 5µsec to 10µsec, and so on.

The second column shows the percentage(x10) of the packets with a PDV within the specific PDV bin range out of the valid number of packets used for the PDV histogram.

The third column shows the number of packets with a PDV within the specific bin range.

The next column is characterized by the letter N. The N stands for Normalized. This column presents number of packets, normalized to a common PDV bin width of 100µsec. This is done in order to compensate for the non-uniform distribution of the bins.

In case of dropped packets, the packet before and after the dropped packet will not be included in the histogram statistics.

The S2M PDV Histogram part is identical to the M2S PDV Histogram.

6.11.3 Frequency Estimation Measurement

Applicability: IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Commands: SetIpcPtpNpmPeriodicFreqEstRep
GetIpcPtpNpmFreqEstRep

The following is an example for frequency estimation report in ppb for an observation time of 60 minutes.

```
=================
Freq Est - 60 Min
-----------------
Min Freq
00  2286
01  2286
02  2286
03  2286
04  2286
05  2286
06  2286
07  2287
08  2287
09  2287
.
.
.
46  2286
47  2286
48  2286
49  2286
50  2286
51  2287
52  2287
53  2287
54  2287
55  2287
56  2286
57  2286
58  2286
59  2286
=================
```

6.11.4 Floor Packet Percentage Report

Applicability: IPC9xxx – BC/Slave modes, IPC17xx Slave mode
Commands: GetIpcPtpNpmFppRep

```
GetIpcPtpNpmFppRep
WinTime:          200
M2SCntM:          6160
M2SCntN:          6160
M2SVPkt:          6160
M2STPkt:          6160
M2SUC:            70
M2SpdvHistNsMax:  110
M2SpercentageM:   1000
M2SpercentageN:   1000
M2SpercentageV:   1000
S2MCntM:          1600
S2MCntN:          1600
S2MVPkt:          1600
S2MTPkt:          1600
S2MUC:            100
S2MpdvHistNsMax:  50
S2MpercentageM:   1000
S2MpercentageN:   1000
S2MpercentageV:   1000
```

6.12 Unicast Tables

6.12.1 Slave

6.12.1.1 GetIpcPtpUnicastMasterTable

Applicability: IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Commands: GetIpcPtpUnicastMasterTable

The following is an example for the unicast master table report.

```
====================================
    Unicast Master Table
====================================
Address            MAC
------------------------------------
192.168.1.21       00:0A:35:01:F4:88
192.168.100.211    00:0A:35:01:E6:2F
------------------------------------
Occupied: 2
Free: 6
====================================
```

The unicast master table includes the following parameters:

Parameter	Description
Address	Master IP address
MAC	Master MAC address

6.12.1.2 GetIpcPtpUnicastAcceptableMasterTable

Applicability: IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Commands: GetIpcPtpUnicastAcceptableMasterTable

The following is an example for the unicast master table report.

```
============================================
         Acceptable Master Table
============================================
Address            MAC               AP1
--------------------------------------------
192.168.1.21       00:0A:35:01:F4:88  128
192.168.100.211    00:0A:35:01:E6:2F  128
--------------------------------------------
Occupied: 2
Free: 6
============================================
```

The acceptable unicast master table includes the following parameters:

Parameter	Description
Address	Master IP address
MAC	Master MAC address
AP1	Override Priority1 of the Announce packet. 0 – No override

6.12.1.3 GetIpcPtpUnicastAcceptableMasterTableExtended

Applicability: IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Commands: GetIpcPtpUnicastAcceptableMasterTableExtended

The following is an example for the unicast master extended table report.

```
================================================================================================
            Acceptable Master Table Extended
================================================================================================
Address       MAC               Alt   PTSF  PTSF  PTSF  PTSF   PTSF   WaitTo  Rx    Rx    Rx     Master
                                Prio. Mode  Total Sync  Ann    Unuse. Restore Ann   Sync  Dresp  Selected
------------------------------------------------------------------------------------------------------------------------------------------------
192.168.1.21  00:0A:35:01:F4:88 0     2     0     0     0      0      60      288   9562  2390   1
192.168.22.2  00:0A:35:01:E6:2F 0     2     1     0     1      1      60      0     0     0      0
------------------------------------------------------------------------------------------------------------------------------------------------
Occupied: 2
Free: 6
================================================================================================
```

The UnicastAcceptableMasterTableExtended includes the following parameters:

Parameter	Description
Address	Master IP address.
MAC	Master MAC address.
Alt Priority1	Override Priority1 in Announce packet. 0 – No override. (alternatePriority1)
PTSF Mode	Indicates Packet Timing Signal Fail (PTSF) mode: 0-OFF (Pass), 1-ON (Fail), 2-AUTO. (ptsfMode)
PTSF Total	Indicates PTSF triggered by one of the following: PTSF Loss Sync, PTSF Loss Announce or PTSF Unusable. (ptsfLossTotal)
PTSF Loss Sync	Indicates Packet Timing Signal Fail (PTSF) for Sync packets or Delay Response packets. i.e. no Sync or Delay Response packets are received from a Master. (ptsfLossSync)
PTSF Loss Announce	Indicates Packet Timing Signal Fail (PTSF) for Announce packets. i.e. no Announce packets are received from a Master. (ptsfLossAnnounce)
PTSF Unusable	Indicates Packet Timing Signal Fail (PTSF) for Unusable. i.e. high PDV.
Wait To Restore	Indicated the Wait To Restore time in minutes. The restore time is the time left in PTSF Total = 1. (ptsfWaitToRestore)
Rx Announce	Indicates received Announce packets counter. (rxAnnCnt)
Rx Sync	Indicates received Sync packets counter. (rxSyncCnt)
Rx Dresp	Indicates received Delay Response packets counter. (rxDrspCnt)
Master Selected	Indicates the selected Master: 0-Not selected, 1-Selected. (masterSel)

6.12.1.4 GetIpcPtpAcceptableTable

Applicability: IPC9xxx
Commands: GetIpcPtpAcceptableTable

This report is applicable for portMapMode!=0.

The following is an example for the acceptableTable report for startMode 0,1,4.

```
==============================
       Acceptable Table
==============================
Address            PI     PN   PS
------------------------------
192.168.1.20        2      2    6
192.168.1.21        1      1    9
```

```
------------------------------
Occupied: 2
Free: 62
==============================
```

The following is an example for the acceptableTable report for startMode 22,23,24.

```
==============================
       Acceptable Table
==============================
Address            PI     PN   PS
------------------------------
00:0A:35:01:F4:88  2      2    6
00:0A:35:01:E6:2F  1      1    9
------------------------------
Occupied: 2
Free: 62
==============================
```

The acceptableTable includes the following parameters:

Parameter	Description
Address	Master address
PI	PortIndex
PN	PontNum
PS	PortState 1 INITIALIZING 2 FAULTY 3 DISBALED 4 LISTENING 5 PRE-MASTER 6 MASTER 7 PASSIVE 8 UNCALIBRATED 9 SLAVE

6.12.1.5 GetIpcPtpBmcKnownMasterTable

Applicability: IPC9xxx – BC/Slave modes, IPC17xx – Slave mode
Commands: GetIpcPtpBmcKnownMasterTable

The following is an example for startMode 1,3,4,8, portMapMode=0, ptpBmcMode=1,2, aptsState=1.

```
================================================================================================================================
                Known Masters Table
================================================================================================================================
 # Address            Parent Port Identity          GM Clock Identity         D  TTL   P1     P2    CC    CA     OS     SR
--------------------------------------------------------------------------------------------------------------------------------
1* 192.168.1.21       00:0A:35:FF:FE:01:F4:15:0002  00:0A:35:FF:FE:01:F4:15   0  9     128    128   6     254    65535  1
2+ 192.168.1.22       00:0A:35:FF:FE:01:F4:16:0100  00:0A:35:FF:FE:01:F4:16   0  999   0      0     248   254    65535  0
--------------------------------------------------------------------------------------------------------------------------------
Occupied: 2
Free: 6
================================================================================================================================
```

The following is an example for startMode 24, portMapMode=0, ptpBmcMode=10, aptsState=0.

```
================================================================================================================================
                Known Masters Table
================================================================================================================================
 # Address            Parent Port Identity          GM Clock Identity         D  TTL   P1     P2    CC    CA     OS     SR
--------------------------------------------------------------------------------------------------------------------------------
1* 00:0A:35:01:F4:21  00:0A:35:FF:FE:01:F4:88:0001  00:0A:35:FF:FE:01:F4:88   0  9     128    128   6     254    65535  1
2  00:0A:35:01:F4:22  00:0A:35:FF:FE:01:F4:89:0004  00:0A:35:FF:FE:01:F4:81   0  8     128    127   248   254    65535  1
--------------------------------------------------------------------------------------------------------------------------------
Occupied: 2
Free: 6
================================================================================================================================
```

The following is an example for startMode 23,24, portMapMode=1, ptpBmcMode=10, aptsState=0.

```
================================================================================================================================================
                Known Masters Table
================================================================================================================================================
 # Address            Parent Port Identity          GM Clock Identity         D  TTL   P1     P2    CC    CA     OS     SR  LP   NS   PI   PN
------------------------------------------------------------------------------------------------------------------------------------------------
1  00:0A:35:01:F4:88  00:0A:35:FF:FE:01:F4:88:0001  00:0A:35:FF:FE:01:F4:88  24  2     128    128   102   254    65535  1   128  0    2    2
2* 00:0A:35:01:F4:89  00:0A:35:FF:FE:01:F4:89:0004  00:0A:35:FF:FE:01:F4:81  24  2     128    127   102   254    65535  1   128  0    3    3
```

```
-------------------------------------------------------------------------------------------------------------------------------------------
Occupied: 2
Free: 6
===========================================================================================================================================
```

The known master table includes the following parameters:

Parameter	Description
#	Master #. The mark “*” indicates the selected master The mark “+” indicates the selected APTS
Address	Parent master IP address (startMode=1,3,4,8,10) Parent master MAC address (startMode=23,24)
Parent Port Identity	Parent port identity
GM Clock Identity	Grand master clock identity
D	Master Domain
TTL	Time To Live measured in sec
P1	Priority 1
P2	Priority 2
CC	Clock Class
CA	Clock Accuracy
OS	Offset Scaled log variance
SR	Steps Removed
LP	LocalPriority. Applicable for ptpBmcMode=10
NS	NotSlave. Applicable for ptpBmcMode=10
PI	PortIndex. Applicable for portMapMode!=0
PN	PortNum. Applicable for portMapMode!=0

6.12.1.6 GetIpcPtpBmcMasterSelectedEx

Applicability: IPC9xxx – BC/Slave modes, IPC17xx – Slave mode
Commands: GetIpcPtpBmcMasterSelectedEx

The following is an example for GetIpcPtpBmcMasterSelectedEx.

```
GM Identity: 00:0A:35:FF:FE:01:F4:15
Parent Identity: 00:0A:35:FF:FE:01:F4:15:0002
Parent Addr: 192.168.1.21
Port Num: 1
Port Index: 1
```

The SelectedMasterEx includes the following parameters:

Parameter	Description
GM Identity	Clock Identity of the grand master. (grandmasterIdentity)
Parent Identity	Port Identity of the parent master. (parentPortIdentity)
Parent Addr	IP/MAC address of the parent master. (parentUniAddr)
Port Num	Port number. (portNum)
Port Index	Port index. (portIndex)

The following is an example for GetIpcPtpBmcMasterSelectedEx in case master was not selected.

```
GM Identity: 00:0A:35:FF:FE:01:F4:16
Parent Identity: 00:0A:35:FF:FE:01:F4:16:0000
Parent Addr: 192.168.0.0
Port Num: 0
Port Index: 0
```

6.12.1.7 GetIpcPtpBmcMasterSelectedUniReport

Applicability: IPC9xxx – BC/Slave modes, IPC17xx – Slave mode
Commands: GetIpcPtpBmcMasterSelectedUniReport

The following is an example for GetIpcPtpBmcMasterSelectedUniReport.

```
=====================================================
            Master Selected Unicast Report
```

```
-----------------------------------------------------
                                Ann       Sync    DelResp
-----------------------------------------------------
logMsgIntervalRequest             1         -5         -3
logMsgIntervalGrant               1         -5         -3
durationFieldRequest            300        300        300
durationFieldGrant              300        300        300
txSigRutsLastMin                  0          0          0
rxSigRutsLastMin                 10         10         10
txSigGrutsLastMin                10         10         10
rxSigGrutsLastMin                 0          0          0
rxPktlogInterval                  1         -5         -5
=====================================================
```

The SelectedMasterUniReportEx includes the following parameters:

Parameter	Description
logMsgIntervalRequest	Request (RUTS) logMsgIntervalRequest
logMsgIntervalGrant	Grant (GRUTS) logMsgIntervalGrant (0 – Deniel)
durationFieldRequest	Request (RUTS) durationFieldRequest
durationFieldGrant	Grant (GRUTS) durationFieldGrant
TxSigRutsLastMin	Number of transmitted REQUEST_UNICAST_TRANSMISSION packets during last minute
RxSigRutsLastMin	Number of received REQUEST_UNICAST_TRANSMISSION packets during last minute
TxSigGrutsLastMin	Number of transmitted GRANT_UNICAST_TRANSMISSION packets during last minute
RxSigGrutsLastMin	Number of received GRANT_UNICAST_TRANSMISSION packets during last minute
rxPktlogInterval	Measured logInterMessagePeriod of the received packets during last minute (Note: 99 – indicates 0 packets).

6.12.2 Master

6.12.2.1 GetIpcPtpAdminSlavesServed

Applicability: IPC9xxx – BC/Master modes, IPC17xx – Master mode
Commands: GetIpcPtpAdminSlavesServed

The following is an example for the GetIpcPtpAdminSlavesServed report.

```
Report: CH=4 AUPR=384 MAUPR=2000 PS=1
CI IP Addr      MAC Addr          Port Identity               T C M TTLs TTLd UpTime   UpET DnTime   DnET PI PN
0  192.168.1.20 00:0A:35:01:F4:08 00:0A:35:FF:FE:01:F4:08:0005 3 1 0  296  296 26:04:31  298 00:00:00    0  2  2
1  192.160.1.22 00:0A:35:01:F4:09 00:0A:35:FF:FE:01:F4:09:0005 1 1 0  299  299 26:05:32  243 00:00:00    0  8  8
2  192.168.10.1 00:0A:35:01:F4:10 00:0A:35:FF:FE:01:F4:10:0002 2 1 0  294  294 26:05:33  242 00:00:00    0  4  4
3  192.168.10.2 00:0A:35:01:F4:11 00:0A:35:FF:FE:01:F4:11:0004 3 1 0  295  295 26:06:03  212 00:00:00    0  5  5
```

The report includes the following parameters:

Parameter	Description
CH	Number of channels (slaves) served
AUPR	Aggregated Unified Packet Rate
MAUPR	Maximal Aggregated Unified Packet Rate x PS
PS	Pre Scale – scale down packet rate factor (e.g. 2 – reduce the packet rate to ½)
CI	Channel Index
IP Addr	Slave IP address
MAC Addr	Slave MAC address
Port Identity	Slave portIdentity address
T	Channel Type – Sync packet rate (1 –128pps, 2 –64pps, 3 –32pps, 4 –16pps, 5 –8pps, 6 – 4pps, 7 –2pps, 8 –1pps, 9 –1/2pps, 10 –1/4pps, 11 –1/8pps, 12 –1/16pps)
C	Commpath state (0 – Down, 1 – Up)
M	Mute – The channel is in mute when the master doesn't get DelayReq or RUTS

	(REQUEST UNICAST TRANSMISSION TLV) packets for the channel from the slave. Applicable for unicast modes only.
TTLs	Time To Live of Sync/FU services measured in sec (MasterChannelReport.timeToLiveSy)
TTLd	Time To Live of DelayResp services measured in sec(MasterChannelReport.timeToLiveDr)
UpTime	Commpath Up Time
UpET	Commpath Up Elapsed Time [sec]
DnTime	Commpath Down Time
DnET	Commpath Down Elapsed Time [sec]
PI	PortIndex. Applicable for portMapMode!=0
PN	PortNum. Applicable for portMapMode!=0

Note *In L2 operation (startMode=22,24), the IP Addr is 0.0.0.0.*

Note *In multicast operation (startMode=2,7,22,24):*

- *CI=0 uses for sending multicast Sync/FU packets. The other CIs used for receiving DelayReq and sending DelayResp packets.*
- *Channel Type (T) of CI=0 is according to the Sync/FU packet rate, The Channel Type of other CIs is 0.*
- *Commpath of CI=0 is always 0. The Commpath of other Cis is according to the link conditions*
- *Mute (M) of all Cis is 0.*

6.12.2.2 GetIpcPtpAnnounceTable

Applicability: IPC9xxx – BC/Master modes, IPC17xx – Master mode
Commands: GetIpcPtpAnnounceTable

The following is an example for a master announce table report.

```
Report: CH=4 AUPR=4 MAUPR=128
 IP Addr       Port Identity                 T R TTL  PI PN A
 192.168.1.20 00:0A:35:FF:FE:01:F4:01:0001  1 1 29   2  2  1
 192.160.1.22 12:34:56:FF:FE:78:9A:BC:0001  1 1 300  8  8  1
 192.168.1.1  11:22:33:FF:FE:44:55:66:0001  2 2 250  4  4  1
 192.168.1.2  11:22:33:FF:FE:44:55:88:0001  2 2 298  5  5  1
```

The announce table includes the following parameters:

Parameter	Description
AUPR	Aggregated Unified Packet Rate – Announce (Group-2, AnnounceTable.AUPR)
MAUPR	Maximal Aggregated Unified Packet Rate – Announce (Group-2, AnnounceTable.MAUPR)
IP Addr	Slave IP address
Port Identity	Slave port identity
T	Channel Type: Log 2 of the granted Announce inter packet period. e.g. -1 – 2pps, 0 – 1pps, 1 – 0.5pps
R	Requested Channel Type: Log 2 of the granted Announce inter packet period. e.g. -1 – 2pps, 0 – 1pps, 1 – 0.5pps
TTL	The time to live before expiration in sec
PI	PortIndex. Applicable for portMapMode!=0
PN	PortNum. Applicable for portMapMode!=0
A	Active: 0 – No Tx Announce, 1 – Tx Announce Applicable for portMapMode!=0

Note *The Master port use the durationField in UNICAST TLV of Announce packets. This durationField use as the Announce TTL.*

6.13 Datasets

6.13.1 GetIpcPtpDsDefault

Applicability: IPC9xxx, IPC17xx, IPC1603
Commands: GetIpcPtpDsDefault

The following is an example for IPC9xxx or IPC17xx – Master mode default dataset.

```
============================================================
                    Default Dataset
------------------------------------------------------------
```

```
Two-step flag.............0
Clock Identity............12:34:56:FF:FE:AB:CD:EF
Number ports .............2
Clock Quality
-------------
   CQ Clock class .........6
   CQ Clock accuracy ......35
   CQ Scaled log variance .65535
Priority 1 ...............128
Priority 2 ...............128
Domain number ............0
Slave only ...............0
Local Priority ...........128
============================================================
```

The following is an example for IPC17xx – Slave mode or IPC1603 default dataset.

```
============================================================
                    Default Dataset
------------------------------------------------------------
Two-step flag.............0
Clock Identity............12:34:56:FF:FE:AB:CD:EF
Number ports .............1
Clock Quality
-------------
   CQ Clock class .........6
   CQ Clock accuracy ......35
   CQ Scaled log variance .65535
Priority 1 ...............255
Priority 2 ...............255
Domain number ............0
Slave only ...............1
Local Priority ...........128
============================================================
```

The default dataset includes the following parameters:

Parameter	Description
Two-step.flag[1]	Denotes if the master clock is transmitting sync (1-step) or sync with follow up (2-step)
Clock Identity	Unique identifier of the clock
Number ports	The number of PTP ports. Master/slave OC have 1 port
CQ Clock class[1,2]	Denotes the traceability of time or frequency that is distributed by the master clock. This parameter is used by the best master clock algorithm (BMCA) to select best master.
CQ Clock accuracy[1,2]	Denotes the accuracy of a master clock
CQ Scaled log variance[1,3]	An estimate of the variations in the local clock from a linear timescale when it is not synchronized to another clock using the protocol.
Priority 1	Used by BMCA to select best master.
Priority 2	Used by BMCA to select best master.
Domain number	Denotes the domain number forming the communication path
Slave only[4]	Denotes the clock is slave only.
Local Priority	Denotes the Local Priority as per G.8275.1.

Note

(1) Applicable when in master mode of operation
(2) Configurable by the application
(3) Optional
(4) Applicable when in slave mode of operation

6.13.2 GetIpcPtpDsCurrent

Applicability: IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603 – BC/Slave mode
Commands: GetIpcPtpDsCurrent

The following is an example for a Current dataset.

```
======================
   Current Dataset
```

```
----------------------
Steps removed........1
Offset From Master..10
Mean Path Delay...3240
======================
```

The current dataset includes the following parameters:

Parameter	Description
Step removed	Denotes the number of communication paths traversed between the local clock and the grandmaster clock.
Offset from master	Denotes the current value of the time difference between a master and a slave as computed by the slave.
Mean path delay	Denotes the current value of the mean propagation time between a master and slave clock as computed by the slave. The mean path delay is the round trip delay divided by 2 in nanoseconds of the last 60 sec.

Note *currentDS shall be updated only if device slaveStateEx as reported by GetIpcPtpState is 8.*

6.13.3 GetIpcPtpDsParent

Applicability: IPC9xxx – BC/Slave modes, IPC17xx – Slave mode, IPC1603
Commands: GetIpcPtpDsParent

The following is an example for a parent dataset.

```
============================================================
                        Parent Dataset
------------------------------------------------------------
Parent
------
   IP/MAC Address…..............192.168.1.21
   Port identity…..............00 :0A :35 :FF :FE :01 :F4 :00 :0001
   Stats.........................0
Observed Parent
---------------
   Offset scaled log variance...65535
   Clock phase change rate......2147483647
Grandmaster
-----------
   Identity.....................00:0A:35:FF:FE:01:F4:00
   Clock Quality
   -------------
      Clock class...............6
      Clock accuracy............254
      Scaled log variance.......65535
   Priority 1...................128
   Priority 2...................128
   Local priority...............128
============================================================
```

The parent dataset includes the following parameters:

Parameter	Description
Parent – IP/MAC Address	IP/MAC address of the parent master.
Parent – Port Identity	Port Identity of the parent master.
Parent – Stats	When set to 1 indicates if the observed parent parameters are valid.
Observed Parent - Offset scaled log variance	An estimate of the variations of the local clock from a linear timescale when it is not synchronized to another clock using the protocol.
Observed Parent - Clock phase change rate	An estimate of the variations of the local clock from a linear timescale when it is not synchronized to another clock using the protocol.

Grandmaster - Clock Identity	Clock Identity of the grand master.
Grandmaster - Clock class	Denote the traceability of the time or frequency distributed by the grandmaster clock. This parameter is used by the best master clock algorithm (BMCA) to select best master.
Grandmaster - Clock accuracy	Denotes the accuracy of the grandmaster clock.
Grandmaster - Scaled log variance	An estimate of the variations of the local clock from a linear timescale when it is not synchronized to another clock using the protocol.
Grandmaster - Priority 1	Used by BMCA to select best master.
Grandmaster - Priority 2	Used by BMCA to select best master.
Grandmaster - Local priority	Applicable for profile-2. Used by BMCA to select best master. Applicable for PtpBmcMode=10.

6.13.4 GetIpcPtpDsTimeProperties

Applicability: IPC9xxx – BC/Master modes, IPC17xx – Master mode
Commands: GetIpcPtpDsTimeProperties

The following is an example for a Time Properties dataset.

```
==========================
  Time Properties Dataset
--------------------------
UTC offset...........34
UTC offset valid.....0
Leap 59..............0
Leap 61..............0
Time Traceable.......1
Frequency Traceable..1
PTP Timescale........1
Time Source..........32
==========================
```

The current dataset includes the following parameters:

Parameter	Description
UTC offset	Denotes the current offset between TAI and UTC if epoch is PTP
UTS offset valid	Denotes if the timePropertiesDS.currentUtcOffset is known to be correct
Leap 59	Denotes that the last minute of the current UTC day contains 59 seconds
Leap 61	Denotes that the last minute of the current UTC day contains 61 seconds
Time Traceable	Denotes of timescale and UTC offset are traceable to a primary reference
Frequency Traceable	Denotes if the frequency determining the timescale is traceable to a primary reference
PTP Timescale	Denotes if the timescale is PTP or arbitrary (ARB)
Time Source	Denotes the source of time used

6.13.5 GetIpcPtpDsPort

Applicability: IPC9xxx, IPC17xx, IPC1603
Commands: GetIpcPtpDsPort

The following is an example for port dataset for slave.

```
============================================================
                      Port Dataset
------------------------------------------------------------
Port Identity….............000A :35FF :FE01 :F48C :0001
Port State..................9
Log Min Delay Req Interval...-5
Peer Mean Path Delay.........0
Log Announce Interval........1
Announce Receipt Timeout.....5
Log Sync Interval............-5
Delay Mechanism..............1
```

```
Log Min Pdelay Req Interval..-5
Version Number...............2
Local Priority ..............128
Not Slave ...................0
============================================================
```

The following is an example for port dataset for master.

```
============================================================
                     Port Dataset
------------------------------------------------------------
Port Identity….............000A :35FF :FE01 :F48C :0001
Port State...................6
Log Min Delay Req Interval...-5
Peer Mean Path Delay.........0
Log Announce Interval........1
Announce Receipt Timeout.....5
Log Sync Interval............-5
Delay Mechanism..............1
Log Min Pdelay Req Interval..-5
Version Number...............2
Local Priority ..............128
Not Slave ...................1
============================================================
```

The port dataset includes the following parameters:

Parameter	Description
Port Identity	The port identity comprised of clock identity and port number
Port State	The port's state
Log Min Delay Req Interval	The maximum delay request packet rate shown as log2 of packet interval
Peer Mean Path Delay	Measured link delay as measured by peer delay request/response (valid only if using P2P delay measurement)
Log Announce Interval	The maximum Announce packet rate shown as log 2 of packet interval
Announce Receipt Timeout	Announce receipt timeout measures in number of announce intervals
Log Sync Interval	The maximum Sync packet rate shown as log 2 of packet interval
Delay Mechanism	Denotes the delay mechanism
Log Min Pdelay Req Interval	The maximum peer delay request packet rate shown as log 2 of packet interval
Version Number	PTP version number
Local Priority	Denotes the LocalPriority as per G.8275.1. Applicable for ptpBmcMode=10
Not Slave	Denotes the NotSlave as per G.8275.1. Applicable for ptpBmcMode=10

6.13.6 GetIpcPtpDsPortReport

Applicability: IPC9xxx, IPC17xx
Commands: GetIpcPtpDsPortReport

The following is an example for the GetIpcPtpDsPortReport.

```
PortDsReport:
PI  Port Identity                   PS   LMDRI PMPD LAI   ARTO LSI  DM   LMPDRI V  LP  NS

01  12:34:56:FF:FE:78:9A:BC:00013  9    -5     0    1    5    -5   1    -5     2  128  0
02  …
03  …
.
.
.
16  …
```

The report includes the following parameters:

Parameter	Description
PI	Port Index

Port Identity	The port identity comprised of clock identity and port number
PS (Port State)	The port's state
LMDRI (Log Min Delay Req Interval)	The maximum delay request packet rate shown as log2 of packet interval
PMPD (Peer Mean Path Delay)	Measured link delay as measured by peer delay request/response
LAI (Log Announce Interval)	The maximum Announce packet rate shown as log 2 of packet interval
ARTO (Announce Receipt Timeout)	Announce receipt timeout measures in number of announce intervals
LSI (Log Sync Interval)	The maximum Sync packet rate shown as log 2 of packet interval
DM (Delay Mechanism)	Denotes the delay mechanism
LPDRI (Log Min Pdelay Req Interval)	The maximum peer delay request packet rate shown as log 2 of packet interval
V (Version Number)	PTP version number
LP (Local Priority)	Denotes the LocalPriority as per G.8275.1. Applicable for ptpBmcMode=10
NS (Not Slave)	Denotes the NotSlave as per G.8275.1. Applicable for ptpBmcMode=10

6.13.7 GetIpcPtpDsPortReportAll

Applicability: IPC9xxx, IPC17xx
Commands: GetIpcPtpDsPortReportAll

The following is an example for the GetIpcPtpDsPortReportAll.

```
PortDsReportAll:
PI   Port Identity                    PS    LMDRI  PMPD  LAI    ARTO  LSI   DM    LMPDRI  V   LP   NS

01   12:34:56:FF:FE:78:9A:BC:00013    9     -5      0     1     5     -5    1     -5      2   128   0
02   …
03   …
.
.
64   …
```

The report includes the following parameters:

Parameter	Description
PI	Port Index
Port Identity	The port identity comprised of clock identity and port number
PS (Port State)	The port's state
LMDRI (Log Min Delay Req Interval)	The maximum delay request packet rate shown as log2 of packet interval
PMPD (Peer Mean Path Delay)	Measured link delay as measured by peer delay request/response
LAI (Log Announce Interval)	The maximum Announce packet rate shown as log 2 of packet interval
ARTO (Announce Receipt Timeout)	Announce receipt timeout measures in number of announce intervals
LSI (Log Sync Interval)	The maximum Sync packet rate shown as log 2 of packet interval
DM (Delay Mechanism)	Denotes the delay mechanism
LPDRI (Log Min Pdelay Req Interval)	The maximum peer delay request packet rate shown as log 2 of packet interval
V (Version Number)	PTP version number
LP (Local Priority)	Denotes the LocalPriority as per G.8275.1. Applicable for ptpBmcMode=10
NS (Not Slave)	Denotes the NotSlave as per G.8275.1. Applicable for ptpBmcMode=10

7 Common Configurations

This section provides common configuration required for the device.

7.1 Initial configuration

7.1.1 IP Address and MAC Address Settings

Applicability: IPC9xxx, IPC17xx, IPC1603

1. Set IP address: **`SetIpcAdminIfcNetIpAddr`**
   a. E.g. **`SetIpcAdminIfcNetIpAddr(0,0,"192.168.10.1")`**
2. Verify IP address: **`GetIpcAdminIfcNetIpAddr`**
3. Set MAC address: **`SetIpcAdminIfcNetMacAddr`**
   a. E.g. **`SetIpcAdminIfcNetMacAddr(0,0,"1a:2b:3c:4d:5e:6f")`**
4. Verify MAC address: **`GetIpcAdminIfcNetMacAddr`**
5. Store in FLASH memory: **`SetIpcAdminStore(0,0)`**
6. Stop: **`SetIpcPtpAdminStop(0)`**
7. Start in the required mode: **`SetIpcPtpAdminStart`**
   a. E.g. sets as slave **`SetIpcPtpAdminStart(0,1,32,0,0)`**

Note *By default, IP address sets to: Slave 192.168.1.20, Master 192.168.1.21, BC 192.168.1.22.*

Note *By default, MAC address sets to: Slave 00:0A:35:01:F4:14, Master 00:0A:35:01:F4:15, BC 00:0A:35:01:F4:16.*

7.1.2 Changing StartMode

Applicability: IPC9xxx, IPC17xx, IPC1603

1. Stop: **`SetIpcPtpAdminStop(0)`**
2. Start in the required mode: **`SetIpcPtpAdminStart`**
   a. E.g. sets as master **`SetIpcPtpAdminStart(0,0,128,0,0)`**
3. Store in FLASH memory: **`SetIpcAdminStore(0,0)`**

Note *By default, IPC9xxx sets to startMode 4. IPC17xx sets to startMode 1.*

7.1.3 Erase configuration from FLASH memory

Applicability: IPC9xxx, IPC17xx, IPC1603

1. Erase configuration from FLASH memory: **`SetIpcAdminStore(0,99)`**

7.2 Profile Setting

7.2.1 IEEE1588 v2 default profile

Setting to IEEE 1588-2008, Annex J - chapter: J.3 Delay Request-Response Default PTP profile. Unicast PTP.

Applicability: IPC9xxx, IPC17xx, IPC1603

1. Erase configuration from FLASH memory (optional): **`SetIpcAdminStore(0,99)`**
2. Stop device: **`SetIpcPtpAdminStop(0)`**
3. Setting configuration startMode:
   master: **`SetIpcPtpAdminStart(0,0,128,0,0)`**
   slave: **`SetIpcPtpAdminStart(0,1,32,0,0)`**
   BC: **`SetIpcPtpAdminStart(0,4,32,0,0)`**
4. Store in FLASH memory (optional): **`SetIpcAdminStore(0,0)`**

Note *By default, IPC9xxx sets to startMode 4. IPC17xx sets to startMode 1.*

7.2.2 ITU-T G.8265.1 profile

Setting to PTP telecom profile for frequency synchronization (ITU-T 8265.1). In this profile, the device synchronize time, phase and frequency.

Applicability: IPC9xxx, IPC17xx, IPC1603

1. Erase configuration from FLASH memory (optional): **`SetIpcAdminStore(0,99)`**

2. Stop device: **SetIpcPtpAdminStop(0)**
3. Setting configuration startMode:
   master: **SetIpcPtpAdminStart(0,0,128,0,0)**
   slave: **SetIpcPtpAdminStart(0,1,32,0,0)**
   BC: **SetIpcPtpAdminStart(0,4,32,0,0)**
4. Setting ITU-T G.8265.1 profile: **SetIpcPtpAdminProfile(0,1)**
5. Store in FLASH memory: **SetIpcAdminStore(0,0)**
6. Reset device: **SetIpcPtpAdminReset(0)**
7. Step #6 above can be replaced by **SetIpcPtpAdminStop(0)**, followed by the proper **SetIpcPtpAdminStart.** (optional)

7.2.3 ITU-T G.8275.1 profile

Setting to PTP telecom profile for time/phase synchronization (ITU-T 8275.1).

Applicability: IPC9xxx, IPC17xx

1. Erase configuration from FLASH memory (optional): **SetIpcAdminStore(0,99)**
2. Stop device: **SetIpcPtpAdminStop(0)**
3. Setting configuration startMode:
   T-GM: **SetIpcPtpAdminStart(0,22,16,0,0)**
   T-TSC: **SetIpcPtpAdminStart(0,23,16,0,0)**
   T-BC: **SetIpcPtpAdminStart(0,24,16,0,0)**
4. Store in FLASH memory (optional): **SetIpcAdminStore(0,0)**

7.3 Slave Port

7.3.1 Adding Master to Unicast Master Table

Applicability: IPC9xxx, IPC17xx, IPC1603

1. Verify master is not part of unicast master table **GetIpcPtpUnicastMasterTable**
2. Add master to unicast master table: **SetIpcPtpUnicastMasterAdd**
   a. E.g. add “192.168.100.10” **SetIpcPtpUnicastMasterAdd(0,”192.168.100.10”)**
3. Store in FLASH memory (optional): **SetIpcAdminStore(0,0)**

7.3.2 Deleting Master from Unicast Master Table

Applicability: IPC9xxx, IPC17xx, IPC1603

1. Check unicast master table **GetIpcPtpUnicastMasterTable**
2. Delete master from unicast master table: **SetIpcPtpUnicastMasterDelete**
   a. E.g. delete “192.168.100.10”
   SetIpcPtpUnicastMasterDelete(0,”192.168.100.10”)
3. Store in FLASH memory (optional): **SetIpcAdminStore(0,0)**

7.3.3 Changing Slave or BC Sync Packet Rate

Applicability: IPC9xxx, IPC17xx, IPC1603

1. Stop: **SetIpcPtpAdminStop(0)**
2. Restart with new sync rate: **SetIpcPtpAdminStart**
   a. E.g. sets as slave, Sync 128pps **SetIpcPtpAdminStart(0,1,128,0,0)**
3. Store in FLASH memory (optional): **SetIpcAdminStore(0,0)**

Note *Changing the slave’s sync rate will cause the slave to lose all timing information.*

Note *Setting the slave’s sync packet rate will automatically set the Sync packet interval requested by the slave in the REQUEST_UNICAST_TRANSMISSION_SIGNALING for Sync.*

7.3.4 Setting Delay Request Packet Rate

Applicability: IPC9xxx, IPC17xx, IPC1603

1. Set the Delay_Req interval: **`SetIpcPtpAdminDelayReqInterval`**
   Set the delay request interval (ptpDelayReqInterval). The ptpDelayReqInterval defines as the ratio between the pktRate and the delayReqRate.
   i.e. ptpDelayReqInterval = pktRate/delayReqRate.
   a. E.g. pktRate = 32 packets per second, desired delayReqRate = 8 packets per second, ptpDelayReqInterval = 4. **`SetIpcPtpAdminDelayReqInterval(0,4)`**
2. Store in FLASH memory (optional): **`SetIpcAdminStore(0,0)`**

Note *Setting the slave's Delay_Req packet rate will automatically set the Delay_Resp packet interval requested by the slave in the REQUEST_UNICAST_TRANSMISSION_SIGNALING for Delay_Resp.*

7.3.5 Manually Setting Slave to Lock to a Master Ignoring UnicastMasterTable

Applicability: IPC9xxx, IPC17xx, IPC1603

1. Ignore UnicastMasterTable using: **`SetIpcPtpBmcIgnoreUniMasterTable(0,1)`**
2. Manually sets the selected Master IP address using: **`SetIpcPtpBmcMasterManSel`**
   a. E.g. select "192.168.1.10" **`SetIpcPtpBmcMasterManSel(0,"192.168.1.10")`**

Note ***`SetIpcPtpBmcMasterManSel`*** *automatically disable BMC.*

7.3.6 Manually Setting Slave to Lock to a Master Using UnicastMasterTable

Applicability: IPC9xxx, IPC17xx, IPC1603

1. Use UnicastMasterTable (default): SetIpcPtpBmcIgnoreUniMasterTable(0,0)
2. Manually sets the selected Master IP address using: **`SetIpcPtpBmcMasterManSel`**
   b. E.g. select "192.168.1.10" **`SetIpcPtpBmcMasterManSel(0,"192.168.1.10")`**
3. Add Master to Unicast Master table using: **`SetIpcPtpUnicastMasterAdd`**
   a. E.g. add "192.168.1.10" **`SetIpcPtpUnicastMasterAdd(0,"192.168.1.10")`**

Note ***`SetIpcPtpBmcMasterManSel`*** *automatically disable BMC.*

Note ***`SetIpcPtpUnicastMasterAdd`*** *is required in case master is not already in Unicast Master table.*

7.4 Master Port

7.4.1 Manually Adding Sync & DelayResp Services

Applicability: IPC9xxx, IPC17xx

1. Set the slave add/release mode to 0: **`SetIpcPtpAdminAutoSlaveAddRelMode(0,0)`**
2. Manually add Slave "192.168.1.11" to CI 0, 32 sync packets/sec:
   a. E.g. **`SetIpcPtpAdminManSlaveAddCi(0,0,"192.168.1.11",-5)`**

Note *It is recommended that the Master will lock to the GPS 1PPS signal prior to adding slaves.*

Note *The first Slave added to the Master should be assigned channel index (CI) 0, for example, SetIpcPtpAdminManSlaveAddCi(0,0,"192.168.1.121",-5) assigns a slave with IP address 192.168.1.121 to channel 0 at 32 sync packets/sec.*

Note *The DelayResp packet rate controlled by the slave.*

Note *As an optional setting, in order to set the slave port, on the other side, not to send RUTS packets for Sync and DelayResp: SetIpcPtpUnicastRutsEn(0,1,0), SetIpcPtpUnicastRutsEn(0,2,0) respectively .*

Note *As an optional setting, in order to set the slave port, on the other side, to ignore UnicastMasterTable: SetIpcPtpBmcIgnoreUniMasterTable(0,1).*

7.4.2 Manually Releasing Sync & DelayResp Services

Applicability: IPC9xxx, IPC17xx

1. Set the slave add/release mode to 0: **`SetIpcPtpAdminAutoSlaveAddRelMode(0,0)`**
2. Manually release Slave in CI: **`SetIpcPtpAdminManSlaveRelCi`**
    a. E.g. release CI=2 **`SetIpcPtpAdminManSlaveRelCi(0,2)`**

Note *If automatic channel add/release is invoked (NcaMode=3,5) then a removed Slave may be automatically added if it continue requesting service from the master.*

7.4.3 Manually Assigning New Sync & DelayResp Services to an Occupied CI

Applicability: IPC9xxx, IPC17xx

1. Set the slave add/release mode to 0: **`SetIpcPtpAdminAutoSlaveAddRelMode(0,0)`**
2. Manually release Slave in CI: **`SetPtpAdminManSlaveRel`**
    a. E.g. release CI=2 **`SetIpcPtpAdminManSlaveRelCi(0,2)`**
3. Manually add Slave to CI: **`SetIpcPtpAdminManSlaveAddCi`**
    a. E.g. add CI=2 **`SetIpcPtpAdminManSlaveAddCi(0,2,"192.168.100.10",-5)`**

7.4.4 Manually Adding Announce Service

Applicability: IPC9xxx, IPC17xx

1. Set the Announce packet transmission mode to manual announce channel allocation: **`SetIpcPtpAnnounceTxMode(0,4)`**
2. Manually add Slave: **`SetIpcPtpAdminAnnounceAdd`**
    a. E.g. add "192.168.1.11", "00:0A:35:FF:FE:01:F4:0B:0001", logPacketPeriod=1 **`SetIpcPtpAdminAnnounceAdd(0,"192.168.1.11","00:0A:35:FF:FE:01:F4:0B:0001",1,2)`**
3. Verify slave was added: **`GetIpcPtpAnnounceTable`**

Note *As an optional setting, in order to set the slave port, on the other side, not to send RUTS packets for Announce: SetIpcPtpUnicastRutsEn(0,4,0).*

Note *As an optional setting, in order to set the slave port, on the other side, to ignore UnicastMasterTable: SetIpcPtpBmcIgnoreUniMasterTable(0,1).*

7.4.5 Manually Releasing Announce Service

Applicability: IPC9xxx, IPC17xx

1. Set the Announce packet transmission mode to manual announce channel allocation: **`SetIpcPtpAnnounceTxMode(0,4)`**.
2. Manually release Slave: **`SetIpcPtpAdminAnnounceDelete`**
    a. E.g. release "192.168.1.11" **`SetIpcPtpAdminAnnounceDelete(0,"192.168.1.11")`**
    b. E.g. release all slaves **`SetIpcPtpAdminAnnounceDeleteAll(0)`**
3. Verify slave/s was/were released: **`GetIpcPtpAnnounceTable`**

Note *If manual announce channel allocation was not set (PtpAnnounceMode=4) then a removed Slave may be automatically added if it continue requesting service from the master.*

7.4.6 Modifying Clock Domain

Applicability: IPC9xxx, IPC17xx

1. Change domain number: **`SetIpcPtpDsDomainNumber`**
    a. E.g. set domain 4 **`SetIpcPtpDsDomainNumber(0,4)`**
2. Store in FLASH memory (optional): **`SetIpcAdminStore(0,0)`**

Note *Modifying the domain number of a Master will cause all slaves currently locked to this master to lose communication path if clock domain will not match the master's clock domain.*

7.5 Configuring Slave to Lock to Master – Default Settings

Applicability: IPC9xxx

1. Verify the two IPC9xxx are in default configuration by doing (optional): **`SetIpcAdminStore(0,99)`**
2. Change StartMode of the IPC9xxx designated to be master:
    1. Stop: **`SetIpcPtpAdminStop(0)`**
    2. Sets as master: **`SetIpcPtpAdminStart(0,0,128,0,0)`**
    3. Store in FLASH memory (optional): **`SetIpcAdminStore(0,0)`**
3. Change StartMode of the IPC9xxx designated to be slave:
    1. Stop: **`SetIpcPtpAdminStop(0)`**
    2. Sets as slave: **`SetIpcPtpAdminStart(0,1,32,0,0)`**
    3. Store in FLASH memory (optional): **`SetIpcAdminStore(0,0)`**
4. Connect the Ethernet cable.

Note *This example can be used in case both devices are in there default settings.*
By default the IPC9000 set to plug & play using unicast negotiation.

7.6 Configuring BC Slave Port to Lock to another BC or Master

Applicability: IPC9xxx

1. Slave side – Add the other BC/Master IP address to unicast master table: **`SetIpcPtpUnicastMasterAdd`**
    a. E.g. add “192.168.100.10” **`SetIpcPtpUnicastMasterAdd(0,"192.168.100.10")`**
2. In case the other BC/Master ClockClass is higher or equal to the BC ClockClass then use **`SetIpcPtpDsClockClass`** to configure adequate ClockClass.
    a. E.g. in case the ClockClass of the other Master is 248, set the BC ClockClass to 249 using **`SetIpcPtpDsClockClass(0,249)`**
3. Store in FLASH memory (optional): **`SetIpcAdminStore(0,0)`**

7.7 Configuring Multicast Slave and Master (UDP)

Applicability: IPC9xxx, IPC17xx, IPC1603

1. Master
    a. Sets as master: **`SetIpcPtpAdminStart(0,7,128,0,0)`**
2. Slave
    a. Sets as slave: **`SetIpcPtpAdminStart(0,8,32,0,0)`**

Note *There is no signaling in multicast. The flow is as follow:*
Master → Announce → Slave
Master → Sync → Slave
Master ← DelayReq ← Slave
Master → DelayResp → Slave.

7.8 Configuring Combined Multicast/Unicast Slave and Master (UDP)

Applicability: IPC9xxx, IPC17xx, IPC1603

1. Master
    a. Sets as master: **`SetIpcPtpAdminStart(0,2,128,0,0)`**
2. Slave
    a. Sets as slave: **`SetIpcPtpAdminStart(0,3,32,0,0)`**

Note *There is no signalling in multicast. The flow is as follow:*
Master → Announce → Slave
Master → Sync → Slave
Master ← DelayReq ← Slave
Master → DelayResp → Slave.

7.9 Configuring Multicast Slave and Master – Modify Packet Rate (UDP)

Applicability: IPC9xxx, IPC17xx, IPC1603

1. Master
   a. Clear FLASH memory (optional): **SetIpcAdminStore(0,99)**
   b. Stop: **SetIpcPtpAdminStop(0)**
   c. Sets as master: **SetIpcPtpAdminStart(0,7,128,0,0)**
   d. Sets Sync packet rate to 16pps: **SetIpcPtpMcastSyncRate(0,-4)**
   e. Store in FLASH memory (optional): **SetIpcAdminStore(0,0)**
2. Slave
   a. Clear FLASH memory (optional): **SetIpcAdminStore(0,99)**
   b. Stop: **SetIpcPtpAdminStop(0)**
   c. Sets as slave: **SetIpcPtpAdminStart(0,8,16,0,0)**
   d. Sets Delay Request packet rate to 16pps: **SetIpcPtpAdminDelayReqInterval(0,1)**
   e. Store in FLASH memory (optional): **SetIpcAdminStore(0,0)**

7.10 Others

7.10.1 Manually Multicast PTP Link Settings

Multicast mode and configuration of the PTP link manually, without Announce packets from the master, and without BMC operation in the slave.

1. Master
   a. Sets as master: **SetIpcPtpAdminStart(0,7,128,0,0)**
   b. **SetIpcPtpAnnounceTxMode(0,0)**
2. Slave
   a. Sets as master: **SetIpcPtpAdminStart(0,10,32,0,0)**
   b. **SetIpcPtpBmcMasterManSel(0,"Master IP address")**

8 Log Message Description

The device provides comprehensive information through the UART management port (mUART). This information is the device's log. The device's log includes information for customers use, as well as information for IPClock's remote support.

The following messages ID number are for customers:

- Message IDs [#00xx], e.g. [#0001], [#0052]
- Message IDs [#01xx], e.g. [#0103]
- Message IDs [#02xx], e.g. [#0201]
- Message IDs [#03xx]
- Message IDs [#11xx]
- Message IDs [#12xx]
- Message IDs [#13xx]

All other message IDs (e.g. Msg [#05xx], [#06xx], [#16xx]) are for IPClock's remote support. They includes information gathered from the device in order to enable IPClock to provide professional remote support to customers. Since this information is for IPClock support only, it doesn't disclosed to customers. Additonal information regarding the message level settings is available in the description of the SetIpcPtpAdminMsgLvl API.

In case the management mode is "Terminal – UART", as set by SetIpcAdminMgmtMode(0,0), which is default setting, the device "pushes" this information to the mUART. In case the management mode is "Host – UART", as set by SetIpcAdminMgmtMode(0,1), the device "pushes" this information to the device internal log buffer that can be read by the host CPU using GetIpcAdminLog API.

In case of SetIpcAdminMgmtMode(0,0), the log information is presented on the PC terminal screen. The user can turn on in the terminal software the option of log session. This way the device log will be redirected to a file as well. This file can be kept by the user as a record.

A log message has the following structure: Time dd:hh:mm:ss, message identification in brackets [#msgid] and message content. The dd:hh:mm:ss is PTP time modulo days, hours, minutes and seconds respectively. The following section provides description of the log messages.

8.1 BMC Messages

Message Content
07:02:19:43 [#0100] BMC: New Master added to bmcKnownMasterTable - 192.168.1.21 # 00:0A:35:FF:FE:01:F4:15:0002 CC 248 PI 2 Indicates that a Master had been added to the bmcKnownMasterTable. (IP addr,portIdentity,clockClass,portIndex)
07:02:19:43 [#0100] BMC: Parent Master Selected - 192.168.1.21 # 00:0A:35:FF:FE:01:F4:15:0002 CC 248 PI 2 Indicate that a Master had been selected as the parent Master. (IP addr,portIdentity,clockClass,portIndex)
07:02:19:43 [#0100] BMC: Master Aged - 192.168.1.21 # 00:0A:35:FF:FE:01:F4:15:0002 CC 248 PI 2 Indicate that a Master had been aged from bmcKnownMasterTable. (IP addr,portIdentity,clockClass,portIndex)
07:02:19:43 [#0100] BMC: Parent Master Deselected Indicate that the selected Master was deselected.
07:02:19:43 [#0100] BMC: Parent Master is inactive - 192.168.1.21 # 00:0A:35:FF:FE:01:F4:15:0002 CC 248 PI 2 Indicates that the parent Master in no longer active. (IP addr,portIdentity,clockClass,portIndex)
07:02:19:43 [#0100] BMC: Updated Foreign Master Dataset – 192.168.1.21 # 00:0A:35:FF:FE:01:F4:15:0002 CC 248 PI 2 Indicates that the properties of a Master in the bmcKnownMasterTable had been updated. (IP addr,portIdentity,clockClass,portIndex)
07:02:19:43 [#0100] BMC: Updated Parent Master Dataset - 192.168.1.21 # 00:0A:35:FF:FE:01:F4:15:0002 CC 248 PI 2 Indicates parent Master properties had been updated. (IP addr,portIdentity,clockClass,portIndex)

Message Content
07:02:19:43 [#0100] BMC: Parent Master Pre-Select - ETI Indicate that ETI had been Pre-select as the parent Master.
07:02:19:43 [#0100] BMC: Parent Master Selected - ETI Indicate that ETI had been selected as the parent Master.
07:02:19:43 [#0100] BMC: Parent Master Pre-De-Select - ETI Indicate that ETI had been pre-de-select as the parent Master.
07:02:19:43 [#0100] BMC: Parent Master De-Selected - ETI Indicate that ETI had been de-selected as the parent Master.

8.2 BC Messages

These messages are applicable to the IPC9xxx – BC mode.

8.2.1 Message Id 1350

Message Content	
07:02:19:43 [#1350] M ST=4 E=8 H=1 S3=3 SG=2 HS=0 M=248 F=1000 1000 P=40 40 n 0 ST:T:LK=21:02:44	
M	Indicates BC Manager message
ST=	Slave state (state as returned by GetIpcPtpState)
E=	Slave state extended (slaveStateEx as returned by GetIpcPtpState)
H=	Holdover ready (0 – Not Ready, 1 – Ready, 2 – Deep Ready, slaveStateHoReady returned by GetIpcPtpStateSlaveHoReady)
S3=	For remote support
SG=	Device status (0 – FAIL, 1 – ALARM, 2 – PASS, gState.status as returned by GetIpcPtpState)
HS=	Hybrid status (Frequency recovered by: 0 - IEEE1588 & SyncE, 1 – SyncE only)
M=	ParentDs.ClockQuality.ClockClass
F=	Floor Packet Percentage x10 (FPP) M2S S2M
P=	PDV [n 0=ns / u 1=us / m 2=ms] M2S S2M
ST:T:	State entry Time (current state only)
FR=	The last time the device entered Free run state
TR=	The last time the device entered Trace state
LK=	The last time the device entered Lock state
HO=	The last time the device entered Holdover state

The following message #1350 applicable for IPC9600 only.

Message Content
07:02:19:43 [#1350] M ST=1 E=1 H=0 S3=1 GST=1 SG=1 HS=0 M=00 F=1000 1000 P=40 40 n 0 ST:T:FR=00:00:00 GST:T:FR=00:00:00

Message Content	
M	Indicates BC Manager message
ST=	Slave state (state as returned by GetIpcPtpState)
E=	Slave state extended (slaveStateEx as returned by GetIpcPtpState)
H=	Holdover ready (0 – Not Ready, 1 – Ready, 2 – Deep Ready, slaveStateHoReady returned by GetIpcPtpStateSlaveHoReady)
S3=	For remote support
GST=	Global status (0 – INIT, 1 – FR, 2 – HO, 3 – TR, 4 – LK, EDStatus.G_ST as returned by GetIpcAdminExtDeviceGlobStatus and GetIpcAdminExtDeviceStatusEx)
SG=	Device status (0 – FAIL, 1 – ALARM, 2 – PASS, gState.status as returned by GetIpcPtpState)
HS=	Hybrid status (Frequency recovered by: 0 - IEEE1588 & SyncE, 1 – SyncE only)
M=	ParentDs.ClockQuality.ClockClass
F=	Floor Packet Percentage x10 (FPP) M2S S2M
P=	PDV [n 0=ns / u 1=us / m 2=ms] M2S S2M
ST:T:	State entry time (current state only)
FR=	The last time the device entered Free run state
TR=	The last time the device entered Trace state
LK=	The last time the device entered Lock state
HO=	The last time the device entered Holdover state
GST:T:	Global state entry time (current state only)
FR=	The last time the device entered Free run state
TR=	The last time the device entered Trace state
LK=	The last time the device entered Lock state
HO=	The last time the device entered Holdover state

8.3 Slave Messages

These messages are applicable to the IPC9xxx – BC/Slave modes, IPC17xx – Slave mode and to the IPC1603.

8.3.1 Message Id 0050

Message Content	
07:02:19:43 [#0050] M ST=4 E=8 H=1 S3=3 SG=2 HS=0 M=00 F=1000 1000 P=40 40 n FR=00:00:00 TR=00:05:51 LK=00:06:01 HO=00:05:51 LK=61741	
M	Indicates Sync Manager message
ST=	Slave state (state as returned by GetIpcPtpState)
E=	Slave state extended (slaveStateEx as returned by GetIpcPtpState)
H=	Holdover ready (0 – Not Ready, 1 – Ready, 2 – Deep Ready, slaveStateHoReady returned by GetIpcPtpStateSlaveHoReady)
S3=	For remote support
SG=	Device status (0 – FAIL, 1 – ALARM, 2 – PASS, gState.status as returned by GetIpcPtpState)
HS=	Hybrid status (Frequency recovered by: 0 - IEEE1588 & SyncE, 1 – SyncE only)
M=	ParentDs.ClockQuality.ClockClass
F=	Floor Packet Percentage x10 (FPP) M2S S2M
P=	PDV [n 1=ns / u 2=us / m 3=ms] M2S S2M
FR=	State entry time: The last time the device entered Free run state
TR=	State entry time: The last time the device entered Trace state
LK=	State entry time: The last time the device entered Lock state
HO=	State entry time: The last time the device entered Holdover state
FR=	State elapsed time (current state only): Free run state elapsed time
TR=	State elapsed time (current state only): Trace state elapsed time
LK=	State elapsed time (current state only): Lock state elapsed time
HO=	State elapsed time (current state only): Holdover state elapsed time

The following message #0050 applicable for IPC9600 only.

Message Content
07:02:19:43 [#0050] M ST=4 E=8 H=1 S3=3 GST=4 SG=2 HS=0 M=00 F=1000 1000 P=40 40 n FR=00:00:00 TR=00:05:51 LK=00:06:01 HO=00:05:51 LK=61741

Message Content	
M	Indicates Sync Manager message
ST=	Slave state (state as returned by GetIpcPtpState)
E=	Slave state extended (slaveStateEx as returned by GetIpcPtpState)
H=	Holdover ready (0 – Not Ready, 1 – Ready, 2 – Deep Ready, slaveStateHoReady returned by GetIpcPtpStateSlaveHoReady)
S3=	For remote support
GST=	Global status (0 – INIT, 1 – FR, 2 – HO, 3 – TR, 4 – LK, EDStatus.G_ST as returned by GetIpcAdminExtDeviceGlobStatus and GetIpcAdminExtDeviceStatusEx)
SG=	Device status (0 – FAIL, 1 – ALARM, 2 – PASS, gState.status as returned by GetIpcPtpState)
HS=	Hybrid status (Frequency recovered by: 0 - IEEE1588 & SyncE, 1 – SyncE only)
M=	ParentDs.ClockQuality.ClockClass
F=	Floor Packet Percentage x10 (FPP) M2S S2M
P=	PDV [n 0=ns / u 1 =us / m 2=ms] M2S S2M
FR=	State entry time: The last time the device entered Free run state
TR=	State entry time: The last time the device entered Trace state
LK=	State entry time: The last time the device entered Lock state
HO=	State entry time: The last time the device entered Holdover state
FR=	State elapsed time (current state only): Free run state elapsed time [sec]
TR=	State elapsed time (current state only): Trace state elapsed time [sec]
LK=	State elapsed time (current state only): Lock state elapsed time [sec]
HO=	State elapsed time (current state only): Holdover state elapsed time [sec]

8.3.2 Message Id 0052

Message Content	
07:02:19:43 [#0052] M C=1 1 1 1 S:Rx=223172 M=0 MO=0 D=0 F:Rx=223172 M=0 D=0 D:Rx=223170 M=0 MO=0 D=0 A:Rx=220 D=0 U:Rx=23040 M:Rx=8	
M	Indicates Sync Manager message
C=	Commpath status (0 – Down, 1 – Up)
1 1 1 or 0 0 0	Used for remote support
S:	Sync packets counters
Rx=	Cumulative number of Sync packets received
M=	Cumulative number of Missed Sync packets (Dropped by the network)
MO=	Cumulative number of Miss ordered Sync packets received
R=	Cumulative number of Sync packets received excluding dropped packets (See note)
D=	Cumulative number of Discarded Sync packets
F:	Follow up packets counters
Rx=	Cumulative number of Follow up packets received
M=	Cumulative number of Missed Follow up packets (Dropped by the network)
D=	Cumulative number of Discarded Follow up packets
D:	Delay response packets counters
Rx=	Cumulative number of Delay Response packets received
M=	Cumulative number of Missed Delay Response packets (Dropped by the network)
MO=	Cumulative number of Miss ordered Delay Response packets received
R=	Cumulative number of Delay Response packets received excluding dropped packets (See note)
D=	Cumulative number of Discarded Delay Response packets
A:	Announce packets counters
Rx=	Cumulative number of Announce packets received
D=	Cumulative number of Discarded Announce packets
U:	Unicast packets counters
Rx=	Total number of Unicast packets received in the last minute
M:	Multicast packets counters
Rx=	Total number of Multicast packets received in the last minute

Note *R counter are presented only in case Signaling 0x0800 (bit 11 set).*

Note *All counters are modulo 1000000.*

Message Content

The follow up counters are presented only in case F:Rx not equal to zero.

Unicast packets counters (U:) and Multicast packets counters (M:) are before PTP domain filtration

8.3.3 Message Id 0150

Message Content	
07:02:19:43 [#0150] M No packets	
M	Indicates Sync Manager message Indicates that the slave is not receiving IEEE 1588 packets

8.3.4 Message Id 0456

Message Content	
07:02:19:43 [#0456] TC T2:E=1 V=10 MN=-10 MX=0 AV=0 AMP=0 BMN=0 T4:E=1 V=20 MN=0 MX=20 AV=10 AMP=0 BMN=0	
TC	Indicates TC E2E
T2:	CF of Sync/FU
E=	Enable/Disable Sync/FU
V=	Last CF Sync/FU value in nsec
MN=	Min CF Sync/FU in nsec during the last minute
MX=	Max CF Sync/FU in nsec during the last minute
AV=	Average CF Sync/FU in nsec during the last minute
AMP=	Above Max Positive - Number of Sync/FU packets during the last minute with CF above max value (slaveCFLimitTh)
BMN=	Below Max Negative - Number of Sync/FU packets during the last minute with CF below min value (slaveCFLimitTh)
T4:	CF of Delay Request (carried by Delay Response)
E=	Enable/Disable Delay Request (carried by Delay Response)
V=	Last CF Delay Response in nsec value
MN=	Min CF Delay Response in nsec during the last minute
MX=	Max CF Delay Response in nsec during the last minute
AV=	Average CF Delay Response in nsec during the last minute
AMP=	Above Max Positive - Number of Delay Response packets during the last minute with CF above max value (slaveCFLimitTh)
BMN=	Below Max Negative - Number of Delay Response packets during the last minute with CF below min value (slaveCFLimitTh)

The message appearance controlled by SetIpcPtpAdminCorrectionFieldReportEn API.

8.3.5 Message Id 0170

Message Content	
07:02:19:43 [#0170] A - Cnt LastCorr AggCorr LastCorrAsymmetry TotalCorr	
A	Indicates Asymmetry Compensation (ACS) Cnt – number of time-updates by AsymComp LastCorr – last asymmetry compensation correction in ns AggCorr – aggregated asymmetry compensation correction in ns LastCorrAsymmetry – asymmetry correction phase (reported by GetIpcAdminPpsOutAsymmetryCorr) TotalCorr – Total asymmetry compensation correction in ns

Message Content
07:02:19:43 [#0170] A – Manual ASC latch was rejected

8.4 Master Messages

These messages are applicable to the IPC9xxx – BC/Master modes and IPC17xx – Master mode.

8.4.1 Message Id 0001

Message Content	
07:02:19:43 [#0001] M ST=3 T:LK=71:20:51 A=0 E1=0 E2=0 U=2	
M	Indicates Master Manager message
ST=	Master state
T:	State entering time (current state only)
FR=	The time the master last enter Free run state
TR=	The time the master last enter Trace state
LK=	The time the master last enter Lock state
HO=	The time the master last enter Holdover state
A=	API utilization counter during the last minute (ApiTotlCnt as reported by GetIpcAdminMgmtPortStat)
E1=	API error (STATUS=-1) counter during the last minute. (ApiStatus-1Cnt as reported by GetIpcAdminMgmtPortStat)
E2=	API busy error (STATUS=-2) counter during the last minute. (ApiStatus-2Cnt as reported by GetIpcAdminMgmtPortStat)
U=	API utilization in %. (max of MPU1 and MPU2 as reported by GetIpcAdminMgmtPortStat)

The following message #0001 applicable for IPC9600 operates in extDeviceMode>40.

Message Content	
07:02:19:43 [#0001] M ST=4 ED 4 -2 GST=4 T:LK=71:20:51 A=0 E1=0 E2=0 U=2	
M	Indicates Master Manager message
ST=	Master state
ED	External device A status -2 – N/A, 0 – INIT, 1 – Reserved, 2 – No signal/No valid signal, 3 – TR, 4 - LK External device B status -2 – N/A, 0 – INIT, 1 – Reserved, 2 – No signal/No valid signal, 3 – TR, 4 - LK
GST=	Global status (0 – INIT, 1 – FR, 2 – HO, 3 – TR, 4 – LK, EDStatus.G_ST as returned by GetIpcAdminExtDeviceGlobStatus and GetIpcAdminExtDeviceStatusEx)
T:	State entering time (current state only)
FR=	The time the master last enter Free run state
TR=	The time the master last enter Trace state
LK=	The time the master last enter Lock state
HO=	The time the master last enter Holdover state
A=	API utilization counter during the last minute (ApiTotlCnt as reported by GetIpcAdminMgmtPortStat)
E1=	API error (STATUS=-1) counter during the last minute. (ApiStatus-1Cnt as reported by GetIpcAdminMgmtPortStat)
E2=	API busy error (STATUS=-2) counter during the last minute. (ApiStatus-2Cnt as reported by GetIpcAdminMgmtPortStat)
U=	API utilization in %. (max of MPU1 and MPU2 as reported by GetIpcAdminMgmtPortStat)

8.4.2 Message Id 0101

Message Content	
07:02:19:43 [#0101] Master Lock	
	Indicates that the Master has locked to the reference clock

8.4.3 Message Id 0102

Message Content	
07:02:19:43 [#0102] Adding new Slave to ch 0: 192.168.1.103	
	Indicate that the master allocated a channel to a Slave. In the example the Master allocated channel 0 to a Slave with IP address 192.168.1.103

8.4.4 Message Id 0103

Message Content	
07:02:19:43 [#0103] Ageing channel 0	
	Indicates that communication path with a slave in channel CI had been down for a period that this CI may be used by another slave if required

8.4.5 Message Id 0201

Message Content
07:02:19:43 [#0201] Cannot allocate new ch (1) Master port failed to add a Slave because there was no free channel to allocate
07:02:19:43 [#0201] Cannot allocate new ch (2) Master port failed to add a Slave because Aggregated Unified Packet Rate (AUPR) exceed Maximal Aggregated Unified Packet Rate (MAUPR)
07:02:19:43 [#0201] Cannot allocate new ch (3) Master port in BC mode failed to add a Slave because Sync packet rate exceed max rate
07:02:19:43 [#0201] Cannot allocate new ch (4,X) Master port failed to add a Slave because requested Sync packet rate exceed limits per channel. While X is the requested logInterMessagePeriod.
07:02:19:43 [#0201] Cannot allocate new ch (5,X) Master port failed to add a Slave because requested DelResp packet rate exceed limits per channel. While X is the requested logInterMessagePeriod.
07:02:19:43 [#0201] Cannot allocate new ch (6) Master port failed to add a Slave because requested 128pps Sync exceed 4 channels.

8.4.6 Message Id 1658

Message Content	
07:02:19:43 [#1658] T RM=5 VM=5 DM=0 CE=0 ID=0 IC=0 UT=0 NMT=0 MT=5 GU=0 UU=0 SQ=0	
RM	Received Messages
VM	Valid Messages
DM	Dropped Messages
CE	Messages with CRC Errors
ID	Messages with Invalid Data
IC	Ignored Characters
UT	Updated TOD
NMT	Not Match TOD
MT	Match TOD
GU	Get UTC failed
UU	Update UTC failed
SQ	Remote support

Note *The message displays the ToD port statistics of the master port over the last 1 minute. It's enabled while uartEn=2 set by SetIpcAdminTodUart.*

Note *NMT counter is been increased only if TOD was not matched however TOD was not updated.*

Note *In order to get the message, higher message level such as SetIpcPtpAdminMsgLvl(0,8191) or SetIpcPtpAdminMsgLvl(0,73471) are required. (8191 = 0x1FFF, 73471 = 0x11EFF)*

Note *RM - Received Messages - number of received NMEA messages during the last minute. VM - Valid Messages - number of valid NMEA messages during the last minute. The master samples the ToD port about every 10-15 second. Thus it's expected that RM will be 4-6. For most of the minutes, VM expected to be identical to RM.*

8.5 Other Messages – IPC9600

8.5.1 Message Id 0460

Message Content	
07:02:19:43 [#0460] E S: E 4 -2 I 4 G 4 SG 2	
E	External device
S:	Status
E	External device A status -2–N/A, -1–Error, 0–INIT, 1–Reserved, 2–No signal/No valid signal, 3–TR, 4–LK (EDStatus.ED_ST as returned by GetIpcAdminExtDeviceStatus and GetIpcAdminExtDeviceStatusEx) External device B status -2–N/A, -1–Error, 0–INIT, 1–Reserved, 2–No signal/No valid signal, 3–TR, 4–LK (EDStatus.ED_STB as returned by GetIpcAdminExtDeviceStatusB and GetIpcAdminExtDeviceStatusEx)

I	Internal status - Slave state 0 – INIT, 1 – FR, 2 – HO, 3 – TR, 4 – LK (gState.state as returned by GetIpcPtpState)
G	Global status 0 – INIT, 1 – FR, 2 – HO, 3 – TR, 4 – LK (EDStatus.G_ST as returned by GetIpcAdminExtDeviceGlobStatus and GetIpcAdminExtDeviceStatusEx)
SG	Device status 0 – FAIL, 1 – ALARM, 2 – PASS (gState.status as returned by GetIpcPtpState)

Note This message applicable for BC and Slave modes while using extDeviceMode 40,42,43,46.

8.5.2 Message Id 0054

Message Content	
07:02:19:43 [#0054] E S:E 4 I 4 G 4 F:Mn 79133 Mx 81562 Av 80375 P 1242 G T:FR=00:00:00 TR=71:19:57 LK=71:20:10 HO=00:00:00 ET:LK=1346	
E	External device
S:	Status
E	External device A status: -2–N/A, -1–Error, 0–INIT, 1–Reserved, 2–No signal/No valid signal, 3–TR, 4–LK (EDStatus.ED_ST as returned by GetIpcAdminExtDeviceStatus and GetIpcAdminExtDeviceStatusEx)
I	Internal status - Slave state: 0 – INIT, 1 – FR, 2 – HO, 3 – TR, 4 – LK (gState.state as returned by GetIpcPtpState)
G	Global status: 0 – INIT, 1 – FR, 2 – HO, 3 – TR, 4 – LK (EDStatus.G_ST as returned by GetIpcAdminExtDeviceGlobStatus and GetIpcAdminExtDeviceStatusEx)
F:	Freq
Mn	Minimal F during the last minute [ppt, 10^-12]
Mx	Maximal F during the last minute [ppt, 10^-12]
Av	Average F during the last minute [ppt, 10^-12]
P	Peak F during the last minute [ppt, 10^-12] max(abs(F:Mn-F:Av),abs(F:Mx-F:Mn))
G	Global status
T:	State entry Time
FR	The last time the device entered Free run state
TR	The last time the device entered Trace state
LK	The last time the device entered Lock state
HO	The last time the device entered Holdover state
ET:	State elapsed time (current state only)
FR	Free run state elapsed time [sec]
TR	Trace state elapsed time [sec]
LK	Lock state elapsed time [sec]
HO	Holdover state elapsed time [sec]

9 Contact Information

IPClock Ltd.
Kiryat Weizman Science Park
Building 13A
P.O. Box 2494 Rehovot 76124
Israel
www.ip-clock.com
info@ip-clock.com

IPCLOCK

InSync

For more information about all IPClock's products
please visit our Web site at: www.ip-clock.com