Agilent IO Libraries Suite

E2094S

Agilent LXI Timing and Events User's Guide

Agilent Technologies

Notices

Manual Part Number

5989-6153

Edition

Third Edition, August 2009

Agilent Technologies, Inc.
815 14th Street SW
Loveland, CO 80537 USA

Trademark Information

Visual Studio is a registered trademark of Microsoft Corporation in the United States and other countries.

Windows NT is a U.S. registered trademark of Microsoft Corporation.

Windows and MS Windows are U.S. registered trademarks of Microsoft Corporation.

Software Revision

This guide is valid for Revisions 15.xx of the Agilent IO Libraries Suite software, where xx refers to minor revisions of the software that do not affect the technical accuracy of this guide.

Warranty

The material contained in this document is provided "as is," and is subject to being changed, without notice, in future editions. Further, to the maximum extent permitted by applicable law, Agilent disclaims all warranties, either express or implied, with regard to this manual and any information contained herein, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. Agilent shall not be liable for errors or for incidental or consequential damages in connection with the furnishing, use, or performance of this document or of any information contained herein. Should Agilent and the user have a separate written agreement with warranty terms covering the material in this document that conflict with these terms, the warranty terms in the separate agreement shall control.

Technology Licenses

The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license.

Restricted Rights Legend

If software is for use in the performance of a U.S. Government prime contract or subcontract, Software is delivered and licensed as "Commercial computer software" as defined in DFAR 252.227-7014 (June 1995), or as a "commercial item" as defined in FAR 2.101(a) or as "Restricted computer software" as defined in FAR 52.227-19 (June 1987) or any equivalent agency regulation or contract clause. Use, duplication or disclosure of Software is subject to Agilent Technologies' standard commercial license terms, and non-DOD Departments and Agencies of the U.S. Government will receive no greater than Restricted Rights as defined in FAR 52.227-19(c)(1-2) (June 1987). U.S. Government users will receive no greater than Limited Rights as defined in FAR 52.227-14 (June 1987) or DFAR 252.227-7015 (b)(2) (November 1995), as applicable in any technical data.

Safety Notices

CAUTION

A **CAUTION** notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in damage to the product or loss of important data. Do not proceed beyond a **CAUTION** notice until the indicated conditions are fully understood and met.

WARNING

A WARNING notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in personal injury or death. Do not proceed beyond a WARNING notice until the indicated conditions are fully understood and met.

Contents

1 Introduction

This *Agilent LXI Timing and Event User Manual* provides an introduction to LXI, describes the Agilent LXI Application Programming Interface (API) and how to use it to develop I/O applications on Microsoft Windows®.

- This chapter includes:
  - LXI Overview
  - Information on Interactive LXI
  - If You Need More LXI Information
- *Chapter 2 - Getting Started with LXI* shows how to build and run a sample program in C/C++ and in Visual Basic.
- *Chapter 3 - LXI Events and Triggering* provides general information about LXI Events and triggering.
- *Chapter 4 - LXI and IEEE 1588 Timing* provides general time scale descriptions and specific information about IEEE 1588 time, synchronization, etc.
- *Appendix A - Test System Applications and Design Considerations* provides general guidelines for developing LXI test systems.
- *Appendix B - LXI API Library Information* provides information on the Agilent API.
- *Glossary* includes major terms and definitions used in this guide.

NOTE

Before you can use the Agilent Interactive LXI API, you must install the Agilent IO Libraries Suite on your computer. See the *Agilent IO Libraries Suite Connectivity Guide* in Chapter 2 for installation instructions.

LXI Overview

LAN eXtensions for Instrumentation (LXI) is a measurement platform based on widely used standards such as Ethernet (IEEE802.3 standard), TCP/IP, Web browsers and IVI drivers. LXI combines the measurement functionality and PC-standard I/O connectivity of standalone instruments with the modularity and compact size of plug-in cards—but without the size or cost of a cardcage.

The following diagram shows a typical LXI system. For most applications where the LXI instruments only need to be synchronized to each other and not necessarily to a civil time source (such as GPS or UTC) the test system PC connects through the intranet to an NTP time server. For applications requiring more precise time management or management to a civil time source, a separate Master Clock may be used. LXI Events are system occurrences that require some type of action. Typically, either the system control program generates the event(s) or an instrument generates the event(s) as a result of some measurement. Two typical LXI events are Timestamping and Triggering.

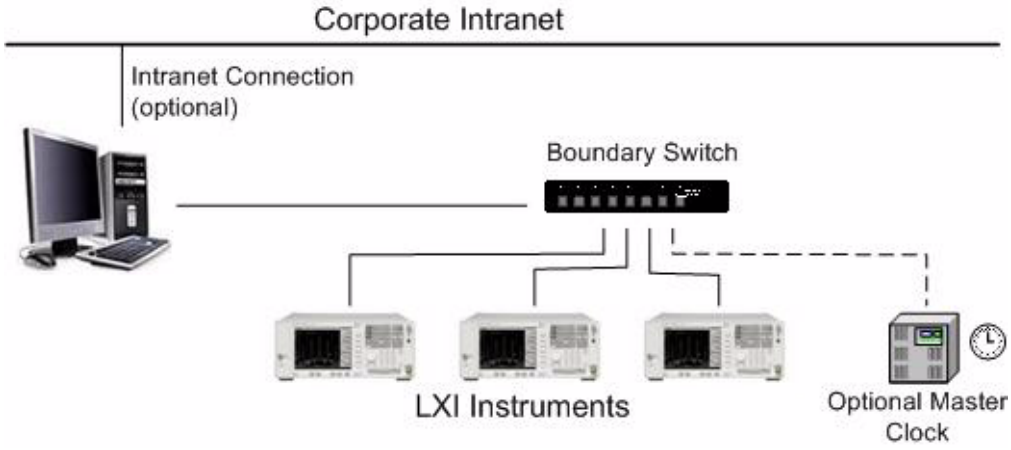


Figure 1 An LXI Test System -- Many will use a regular router instead of a Boundary Switch

LAN is at the heart of LXI. However, instead of modifying existing standards, LXI specifies the interaction of proven standards in five areas: physical, Ethernet, programmatic interface, instrument pages and synchronization.

Physical

To achieve physical consistency, the LXI standard begins with IEC-standard rack dimensions. To help simplify system integration and implementation, it also recommends the placement of various connections. For example, compliant instruments use the front panel for signal inputs and outputs plus indicator lights for LAN, power and IEEE 1588 (synchronization). The rear panel is used for hardware triggering, power input and Ethernet communication. Each LXI module must meet worldwide standard EMI shielding and cooling specifications.

Form factor

To serve a wide range of requirements, LXI devices can be implemented in a variety of form factors. At present, the most common form will be classic benchtop instruments that include a front-panel display and user interface. For computer- controlled systems in manufacturing or aerospace/defense applications, there is a compact, faceless form-factor that enables small, cost effective systems well-suited to local or remote deployment. These modules will typically be 1U to 4U high and half- or full-rack wide. The LXI standard also makes provisions for devices such as sensors, amplifiers, filters and attenuators in compact enclosures that can be deployed inside test fixtures and in remote locations.

Leverage

Although not mentioned in the standard, LXI enables leverage from classic instruments into faceless modular instruments and synthetic instruments (SIs). Agilent is already moving in this direction with the introduction of SIs based on popular benchtop RF and microwave products. By using the same measurement hardware in both classic and modular instruments, we’re boosting your ability to leverage test software as the system evolves.

Ethernet

LXI uses the IEEE 802.3 networking standard to define the appropriate connections, protocols, speeds, addresses, configuration and default conditions that must be implemented to ensure a consistent — yet easy-to use — test system.

- **Connections:** LXI devices use standard RJ-45 connectors and implement Auto-MDIX to sense the polarity of LAN cables (through or crossover).
- **Protocols:** Compliant devices are required to implement TCP (transmission control protocol), UDP (user datagram protocol) and IPv4 (Internet protocol version 4). TCP is the standard Internet protocol that will be used most often in peer-to-peer messaging while UDP is a low-overhead protocol that will be typically used for multicast messaging when high speed delivery is critical.
- **Speeds:** The standard recommends use of 1-Gb Ethernet (and permits 100-Mb) with auto-negotiation to ensure that devices use their optimum speed.
- **Addresses:** LXI devices must support IP addresses (assigned by the server), MAC addresses (assigned by the manufacturer) and hostnames (assigned by the user).
- **Configuration:** Compliant devices must support ICMP (ping server), DHCP-based assignment of IP addresses, manual Domain Name Server (DNS) and Dynamic DNS. Because DNS can translate domain names into IP addresses, it can contribute to the longevity of system software: IP addresses may change but domain names will not.
- **Default conditions:** As a safeguard, LXI defines a set of default LAN conditions and requires a “LAN configuration initialize” (LCI) switch that will reset a device to this set of known conditions.

Programmatic Interface

Because the LXI standard requires that all devices have an Interchangeable Virtual Instrument (IVI) driver, it allows you to use whichever programming language or development environment you prefer. IVI-COM and IVI-C are well-established industry standard drivers that instrument makers supply with their products.

LAN Discovery

The LXI standard mandates that compliant devices implement LAN discovery, which enables the host PC to identify connected instruments. Currently, the standard requires use of the VXI-11 protocol, which defines LAN-based connectivity for all types of test equipment, not just VXI. Going forward, future revisions to the LXI standard may include other discovery mechanisms such as Universal Plug&Play (UPnP).

Instrument Web Pages

LXI-compliant devices must serve their own Web page. This page provides key device information including its manufacturer, model number, serial number, description, hostname, MAC address and IP address. The standard requires a browser-accessible configuration page to change parameters such as hostname, description, IP address, subnet mask and TCP/IP configuration mode. To accessing the Web page, type the instrument IP address into the address line of any browser.

Many of Agilent's LXI-compliant instruments go beyond the LXI requirements, providing monitor and control capabilities. For example, you can set up a DMM, command it to start making measurements and then read the results. Some of our LXI devices even allow you to download complete measurement personalities — CDMA, GSM, Wi-Fi — into the instrument and perform specific measurements with one command.

NOTE

In some network configurations, a proxy server cannot be used to access the instrument IP addresses. In these situations, the browser must be set to disable the proxy for the instrument's address; i.e., **Tools>Internet Options>Connections>LAN Settings>Advanced>Exceptions...**

Synchronization

One especially intriguing aspect of LXI is its triggering and synchronization capabilities. By harnessing the capabilities of LAN and the IEEE 1588 time-synchronization protocol, LXI provides a variety of triggering modes that are not available in other measurement platforms

LXI Device Classes

The LXI standard defines three types of instruments that you can easily mix and match within a test system.

Class C

These are standalone or bench-type instruments that replace GPIB with LAN and harness the full breadth of LAN's capabilities. They also support a Web interface for instrument set-up and data access viewable from a standard web browser. To simplify programming, Class C instruments provide an IVI driver API (application programming interface). This class of instruments supports traditional triggering methods (similar to a GPIB Group Execute Trigger), but does not include any LXI triggering extensions.

Class B

These devices are designed to enable distributed measurement systems. They meet Class C requirements and include IEEE 1588 Precision Time Protocol and Event Triggering. This makes it possible to achieve sub-microsecond synchronization of LXI devices located anywhere on a local network. This class of instruments can use time-based triggers and LAN-based triggers. Class B also adds peer-to-peer and multicast LAN messaging (required in Class B and A, permitted in Class C).

Class A

Devices in this category satisfy Class C and B requirements and adds a fast Hardware Trigger Bus (See “LXI Hardware-based Triggering” on page 44). The trigger bus, similar to the VXI backplane trigger bus, is an eight-line, differential-voltage bus that enables 5nsec/meter timing accuracy for local instruments.

With IEEE 1588 clock synchronization, Class A and B instruments provide the ability to use time-based triggering functions and to respond to triggers on the LAN. Class A instruments add a special programmable hardware bus, similar to the triggering available in VXI instruments, that can be used when time-based or LAN-based triggering is not needed.

LXI Device-to-Device Communication

Standard LXI data packets are either multicast on the LAN via UDP or transmitted through a point-to-point TCP connection. Each message includes a time stamp to indicate a system event. Instruments in the system can be programmed to broadcast messages (or not) as needed.

This message format is required for all LXI Class A and Class B devices. Device-to-device communication is not required in Class C devices. However, Class C devices may participate in device-to-device communications by using a zero value for the time stamp.

LXI device-to-device communications use UDP/TCP port number 5044. For UDP multicasting, data packets are transmitted to the UDP multicast address 224.0.23.159.

Data Packet Format

Figure 2 Shows the device-to-device data packet format.

HW Detect (3 bytes)	Domain (1 byte)	Event ID (16 bytes)	Sequence (UInteger32)	IEEE 1588 Time stamp (10 bytes)	IEEE 1588 Epoch (UInteger16)	Flags (Uinteger16)	Data Fields...	0 (2 bytes)

Figure 2 LXI Device-to-Device Data Packet Format

Hardware Detect: Identifies valid packets, and also reserved for future hardware detection of Event Manager packets. This field should be set to the value “LXI.” Note that the third byte, ASCII “I”, is also used as a version identifier; future revisions to the LXI standard may change this value.

Domain: The default value is 0 (zero). It is treated as an unsigned byte.

Event ID: This field contains the first 16 bytes of the event name (a string) specified in the LXI API. Some event names are pre-defined by the LXI Consortium (e.g., LAN0 - LAN7). All other names are available for use by users.

Sequence: Each instrument maintains its own sequence, and the sequence number is incremented every time a unique packet is transmitted. (Note: If packets are re-transmitted to enhance reliability, re-transmitted packets contain the same sequence number as the original.)

IEEE 1588 Time Stamp: A time stamp identifies the time that the event occurred. If no event timestamp is available, for example if the event is derived from an LXI Class C device incapable of assigning a timestamp, a time value of 0 (zero) is used. A value of 0 for a timestamp is interpreted as "now," i.e., the time when the recipient handles the message.

IEEE 1588 Epoch: The IEEE 1588 epoch.Currently, this is 0 (zero).

Flags: Bits within the flag byte are defined as follows:

- **Bit 0 – Error Message**: If set to 1, indicates that this packet is an error message.
- **Bit 1 – Retransmission**: If set to 1, indicates that this packet is a re-transmission of a prior packet and contains identical information. Retransmitted packets must use the same Sequence number as the original transmission. Retransmission allows devices to transmit the same packet multiple times (for increased reliability), if desired. LXI devices are not required to implement this feature and ignore packets if this bit is set.
- **Bit 2 – Hardware Value**: A logical value that characterizes trigger events (particularly hardware events). Refer to the programmatic interface section of the LXI Standard for further explanation. If not used, this bit is set to zero.
- **Bit 3 – Acknowledgement:** If set to 1, indicates that this packet is an acknowledgement that a prior packet was successfully received. This allows LXI systems to implement UDP-based handshaking protocols (for increased reliability), if desired. Modules are not required to implement this feature; however, those modules shall ignore packets it this bit is set. Bits 4-15 are reserved and are set to zero.

Data Fields: Arbitrary number of bytes, up to the capacity of the data packet. Each data field is formatted as follows:

- ***Data Length*** **(UInteger16):** Length of the data that follows. This field contains a zero if no further data is contained in the packet.

- ***Identifier* (Integer8):** A user-definable identifier that specifies the type of data to follow. Numbers from zero to 127 are available for user-defined identifiers.
- ***User Data* (succeeding bytes):** Data as an octet-array given by the *Length* field. The length value does not include the identifier field itself.

There may be multiple data fields in an Event Manager packet. The packet ends when a zero is encountered (two bytes) as the length of the next field, or when the maximum data payload limit is reached.

This variable-length data field satisfies two different needs. First, it allows the LXI Consortium to define new data fields that may become a part of the LXI specification. Second, it allows vendors to define proprietary data fields of their own. Such proprietary data fields would naturally not be understood by modules from other vendors and should be ignored.

Firewalls.

Unfortunately, firewalls cannot distinguish LXI and IEEE 1588 requests for incoming connections (for LXI events and 1588 messages) from other, similar requests. You will need to configure your firewall utility to allow incoming connections on the LXI port (5044) and the 1588 ports (typically 319 and 320).

Interactive LXI

Agilent's Interactive LXI utility provides a simple interface for testing, monitoring, and generating LXI events, and IEEE 1588 (PTP) Timing Messages. Interactive LXI consists of two tools, the Events Tab and the Timing Tab.

NOTE

Firewalls may block LXI and PTP LAN traffic.When starting Interactive LXI (or any application using the LXI or PTP library), an active Firewall on the PC may block access to the standard LXI and PTP network ports.

To solve this problem, set the Firewall configuration to allow connections to network ports 5044 (LXI), 319 (PTP), and 320 (PTP) either for the specific application or all applications.

The Interactive LXI tool is available by selecting **Interactive LXI** from the IO Libraries Control Icon (under **Utilities**), or from the Windows **Start** button, select **All Programs>Agilent IO Libraries Suite>Utilities>Interactive LXI.** Interactive LXI has two control tabs: **Events** and **Timing**.

Menu

Under the **File** menu, you can select the network adapter you are using in your system. Note that changing the adapter closes all active LXI send and receive connections. Note that the title bar of the Interactive LXI GUI changes to indicate which Network Adapter you specified.

The **Tools** menu returns you to Agilent's Connection Expert.

Interactive LXI Events

Interactive LXI provides two tools for understanding and debugging LXI Events.

- **LXI Event Send**, generates LXI UDP-based or TCP-based trigger messages.
- **LXI Event Receive**, monitors/displays LXI UDP-based trigger messages and TCP-based messages if the port is included in the message; true peer-to-peer TCP-based messages are not displayed. It also indicates which Network Adapter you specified.

For information on using the Event Tab (Figure 3), refer to the Interactive LXI Help.

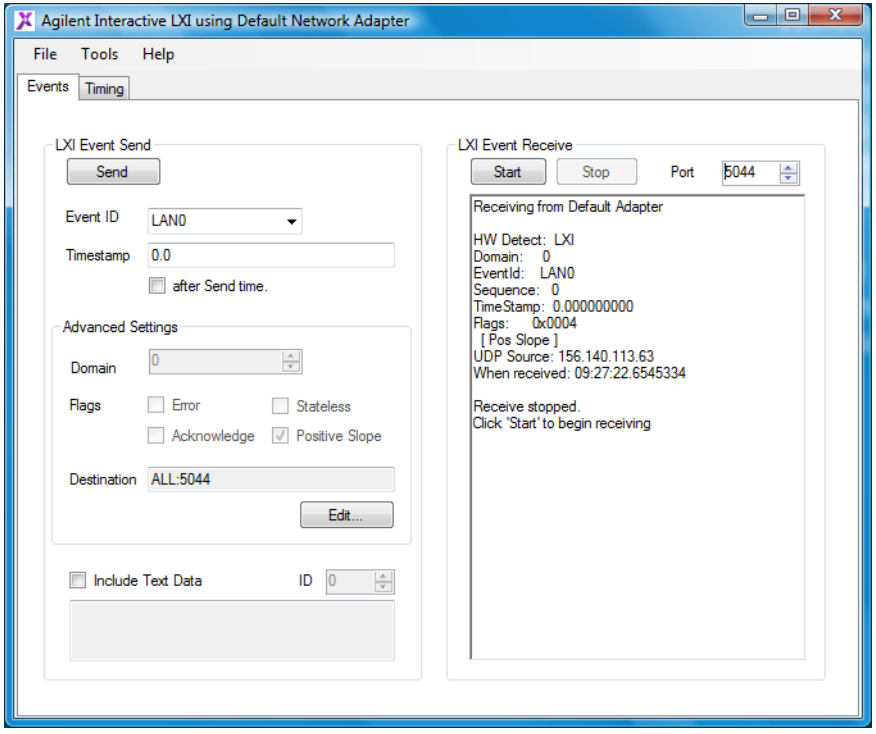


Figure 3 Agilent's Interactive LXI (Event Tab screen)

Interactive LXI Timing

Agilent's Interactive LXI Timing Tab provides clock monitoring and management functionality. Such information includes: Master/Slave Clock identifier, raw IEEE 1588 time and local time, sync configuration, etc. The Interactive LXI Timing Tab provides several tools for understanding your IEEE 1588 implementation.

For information on using the Timing Tab (Figure 4), refer to the Interactive LXI Help file.

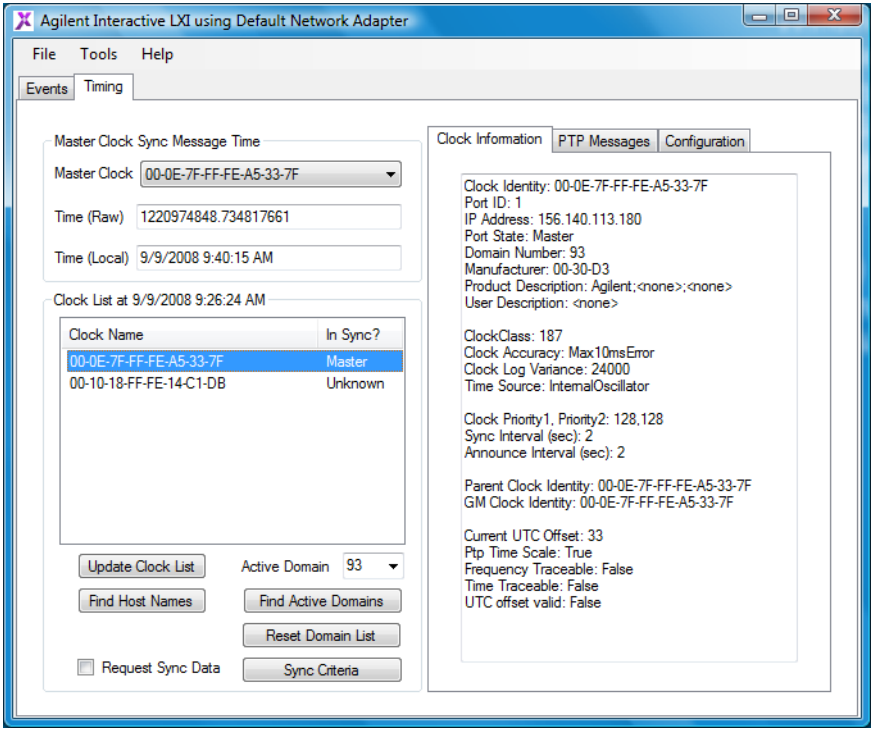


Figure 4 Agilent's Interactive LXI (Timing Tab screen)

If You Need More LXI Information

- In the USA, you can reach Agilent Technologies at this telephone number:

  USA: 1-800-829-4444

- Outside the USA, contact your country's Agilent support organization. A list of contact information for other countries is available on the Agilent web site:

  http://www.agilent.com/find/assist

- The Agilent System Developer Center:

  http://www.agilent.com/find/systemcomponents

  is a one-stop Web resource that supports your connectivity needs with software downloads, sample code, technical notes and white papers.

- For additional information about LXI-compliant products, as well as licensing, specifications and consortium membership refer to:

  http://www.lxistandard.org

  The consortium is a not-for-profit corporation. Its primary purpose is to promote the development an d adoption of the LXI standard as an open, accessible standard that identifies specifications and solutions relating to functional test, measurement, and data acquisition industries.

- For information on the IEEE 1588 Precision Time Protocol, refer to:

  http://ieee1588.nist.gov/

1 Introduction

2
Getting Started with LXI Programming

Several LXI Event and IEEE 1588 PTP example programs are available as part of the IO Libraries Suite 15 typical installation. Refer to the directory:
`C:\Program Files\Agilent\IO Libraries Suite\Programming Samples\`

The C#, C, and VB6 example programs with the same names implement the same functionality.

This chapter highlights two sample programs, one in C# (pronounced "C Sharp") programming language and one in Visual BASIC (VB6) programming language. Additional sample programs are available in C, C#, and VB6 programming languages.

The chapter contents are:

- Getting Started Using C#
  - Linking to Agilent's VISA Header File
  - C# Sample - LxiEventManagerSampleApp
- Getting Started Using Visual Basic
  - Linking to Agilent's VISA Header File
  - Visual Basic Sample - PtpSyncLogger.bas

NOTE For additional information, refer to the Agilent VISA User's Guide available as part of the typical installation of your IO Libraries Suite.

Adapting the Sample Programs

C# Programs

- If the install location for IO Libraries Suite is in a different location than the default ("C:\Program Files\Agilent\IO Libraries Suite"), the reference to LxiEventManager or PtpManager may need to be re-entered using the new location.

C Programs

- If VISA is not installed in the default location, the new location of the 'include' directory for VISA will need to be added to allow finding Visa.h.
- If IO Libraries Suite is not installed in the default location, the new location of the 'include' directory for IO Libraries Suite will need to be added to allow finding LxiEventManagerC.h.
- If IO Libraries Suite is not installed in the default location, the new location of the 'lib' for IO Libraries Suite will need to be added to allow finding LxiEventManagerC.lib.

VB6 Programs

- The LxiEventManagerC.bas or PtpManagerC.basmodule must be added to the example. The starting comments in the sample application programs describe how to do this.

Getting Started Using C#

C# is a powerful programming language similar to, but more powerful and flexible than C++. C# programs end with the `.CS` extension. It is Microsoft's flagship programming language on their Visual Studio .NET Framework (Visual C# is the C# component of Visual Studio).

The example programs installed with Agilent's IO Libraries Suite include C# source files (with accompanying Visual Studio project and solution files) demonstrate the use of Agilent's LXI Management API commands.

C# Sample - LxiEventManagerSampleApp.cs

The LxiEventManagerSampleApp sample program creates a simple user interface (see Figure 5 below) for sending/receiving LXI Events. Note that the program uses a UDP broadcast of the event. The time is not a true IEEE 1588 time, it is only an approximation for demonstration purposes. The program does not synchronously receive events.

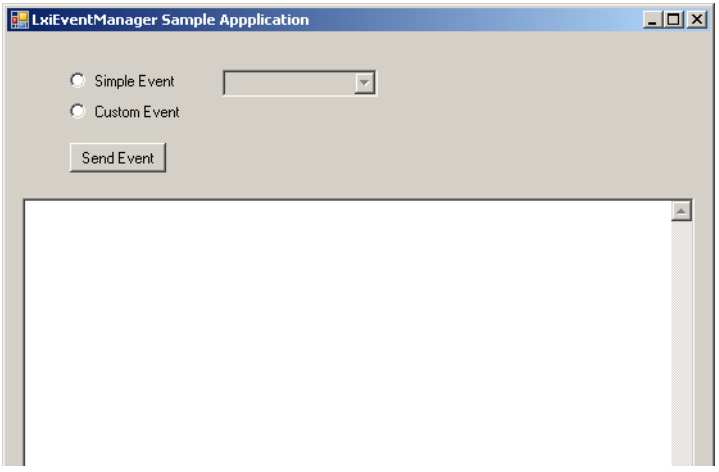


Figure 5 Sample Program Graphical User Interface

NOTE Comment lines in the following program example have been reformatted from what appears in the actual file. This allows for a better presentation in this manual.

C# Sample Program Code

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using Agilent.TMFramework.Lxi.Event;

namespace LxiEventManagerSampleApp
{
    public partial class LxiEventManagerSampleApp : Form
    {

        public LxiEventManagerSampleApp()
        {
            InitializeComponent();
        }

// This is the LxiEventManager used by this application
        private LxiEventManager m_lxiEventManager;

        private void LxiEventManagerSampleApp_Load(object
sender, EventArgs e)
        {
// Populate EventIdComboBox with standard LXI event ID's.
            EventIdComboBox.Items.Clear();
            EventIdComboBox.Items.Add(LxiEventId.LAN0);
            EventIdComboBox.Items.Add(LxiEventId.LAN1);
            EventIdComboBox.Items.Add(LxiEventId.LAN2);
            EventIdComboBox.Items.Add(LxiEventId.LAN3);
            EventIdComboBox.Items.Add(LxiEventId.LAN4);
            EventIdComboBox.Items.Add(LxiEventId.LAN5);
            EventIdComboBox.Items.Add(LxiEventId.LAN6);
            EventIdComboBox.Items.Add(LxiEventId.LAN7);
```

```
// Create LxiEventManager & sign up for EventReceived event
            m_lxiEventManager = new LxiEventManager();
            m_lxiEventManager.EventReceived += new LxiEven-
tReceivedHandler(m_lxiEventManager_EventReceived);

// Start monitoring events received on the default LXI port
via UDP multicast or TCP/IP unicast to this PC.
// This application will see all locally-visible LXI Events,
including those sent by this application.
            m_lxiEventManager.StartReceiving();
        }

        void m_lxiEventManager_EventReceived(object sender,
LxiEventReceivedEventArgs data)
        {
            if (data != null && data.LxiEventData != null)
            {
// Add event summary to EventTextBox and scroll to the bottom
                EventTextBox.SuspendLayout();
              EventTextBox.Text += String.Format("{0}\r\n",
data.LxiEventData.ToString());
                EventTextBox.SelectAll();
                EventTextBox.ScrollToCaret();
                EventTextBox.ResumeLayout();

// NOTE: the summary contains the sending endpoint and
received time stamps, but the 'data' parameter also contains
these properties should this application want to use that
information programmatically
                string eventSender = data.RemoteEnd-
Point.ToString();
                string receivedTimeStamp = data.Received-
TimeStamp.ToLocalTimeString();
                bool cameViaUdp = data.ReceivedViaUdp;
            }
        }

       private void SimpleRadioButton_CheckedChanged(object
sender, EventArgs e)
        {
            EventIdComboBox.Enabled = SimpleRadioBut-
ton.Checked;
        }

        private void SendEventButton_Click(object sender,
```

```
EventArgs e)
        {
            if (SimpleRadioButton.Checked)
            {
// Send a simple event, setting only the EventID The event is
sent to all local instruments via UDP using the default Lxi
address and port.
                m_lxiEventManager.SendEvent(EventIdCom-
boBox.Text);
            }
            if (CustomRadioButton.Checked)
            {
// Create a custom event
                LxiEventData lxiEvent = new LxiEventData();

// Set the heading for an error Lxi event.
// (Generally, LxiEventData should use the defaults for
Heading.Domain and Heading.HWDetect)
// The standard LXIERROR id should always have Error flag set
                lxiEvent.Heading.EventID = LxiEventId.LXIER-
ROR;
               lxiEvent.Heading.Flags = LxiEventFlags.Error;

// Get the next sequence number for the default LXI destina-
tion and domain (each destination path + domain combination
is required to use incrementing sequence numbers)
                lxiEvent.Heading.Sequence =
m_lxiEventManager.GetNextLxiSequenceId(
                    DestinationPath.DefaultDestinationPath,
                    LxiEventData.DefaultDomain);

// Set the data fields (most instrument events won't contain
these, and most instruments will ignore these)
               LxiData dataField = new LxiData(new byte[] {
0, 1, 2 }, 255);
                lxiEvent.Data = new LxiData[1];
                lxiEvent.Data[0] = dataField;

// Set the timestamp to 'now' as the PC understands it
(Unless there is a 1588 clock on the PC that LxiEventTime
knows about, this won't be a 1588-synchronized time)
                lxiEvent.Time = LxiEventTime.Now;

// Send the custom event to the default LXI address and port.
There are other SendEvent overloads that allow specifying
```

```
the destination
                m_lxiEventManager.SendEvent(lxiEvent);
            }
        }
    }
}
```

Getting Started Using Visual Basic

This section provides guidelines to getting started programming applications in Visual Basic 6.0 (VB 6.0).

Note that there are a few VISA limitations in Visual Basic. For example, you cannot use the ON ERROR construct in VB or the value of the VB Error variable to catch VISA function errors.For more information, refer to the Agilent VISA User's Guide.

Linking to Agilent's VISA Header File

You must link (or copy the file to your project directory) the VISA32.bas header file to your Visual Basic project in order for it to work properly. To link the file:

1. Right-click the project you wish to modify (not the solution) in the Solution Explorer window of the Microsoft Visual Studio environment.
2. Click **Add** and then click **Add Existing Item...**
3. Navigate to **C:\Program Files\VISA\Winnt\include\visa32.bas** file to link to in your project. Select it, but do not click the **Open** button.
4. Click the down arrow to the right of the **Open** button, and choose **Link File**.
5. You should now see the file underneath your project in the Solution Explorer. It will have a little arrow icon in its lower left corner, indicating that it is a link.

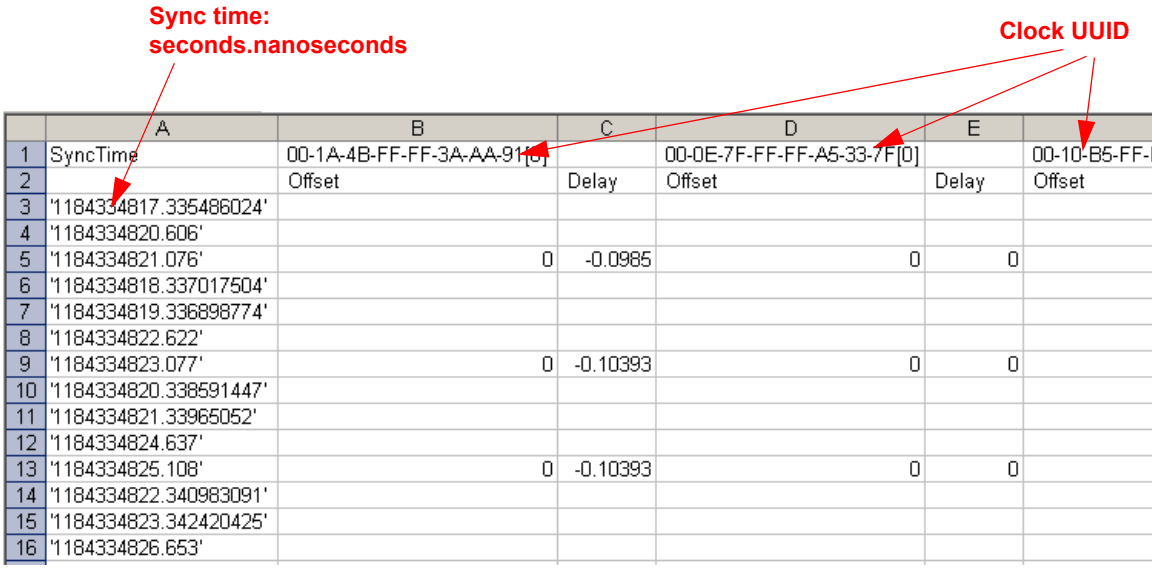
For additional information on using the `VISA32.bas` header file in your Visual Basic programs, refer to the VISA help file.

NOTE Some comment lines in the following program example have been removed or reformatted from what appears in the actual file. This allows for a better presentation in this manual.

Visual Basic Sample - PtpSyncLogger.bas

This program finds active clocks in the local subnet, then logs synchronization history to a file in a comma-separated-variable format compatible with Microsoft Excel. It demonstrates how to monitor PTP clocks and receive PTP timing messages. A sample output file is also provided in the VB6 Sample Programs folder.

Figure 6 shows part of a sample output file.



	A	B	C	D	E	
1	SyncTime	00-1A-4B-FF-FF-3A-AA-91[0]		00-0E-7F-FF-FF-A5-33-7F[0]		00-10-B5-FF-
2		Offset	Delay	Offset	Delay	Offset
3	'1184334817.335486024'					
4	'1184334820.606'					
5	'1184334821.076'	0	-0.0985	0	0	
6	'1184334818.337017504'					
7	'1184334819.336898774'					
8	'1184334822.622'					
9	'1184334823.077'	0	-0.10393	0	0	
10	'1184334820.338591447'					
11	'1184334821.33965052'					
12	'1184334824.637'					
13	'1184334825.108'	0	-0.10393	0	0	
14	'1184334822.340983091'					
15	'1184334823.342420425'					
16	'1184334826.653'					

Figure 6 Partial PtpSyncLogger Output file Example

Visual Basic Program Sample Code

```
Option Explicit


'' To develop PtpManagerC applications in Microsoft Visual
'' Basic, you first need to add the Visual Basic (VB)
'' declaration file in your VB project as a Module. This file
'' contains the PtpManagerC function definitions and
'' constantdeclarations needed to make PtpManagerC calls
'' from Visual Basic.
'' To add this module to your project in VB 6, from the menu,
'' select Project->Add Module, select the 'Existing' tab,
'' and browse to the directory containing the VB Declaration
'' file, select PtpManagerC.bas, and press 'Open'.
''
'' The location of the VB declaration file depends on where
'' IO Libaries Suite is installed. Assuming the default IO
'' Libraries Suite install point this file can be found at:
''
''  C:\Program Files\Agilent\IO Libraries Suite\Include\
PtpManagerC.bas
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

'  PtpSyncLogger.bas
'
' This program finds the active clocks, then logs the
synchronization history for those clocks to stdout.
' The format of the output is a comma-separated-value format
suitable for consumption by Excel.
'
'  "SyncTime", <ClockUUID1>,, <ClockUUID2>,, ...
'  , "Offset", "Delay", "Offset", "Delay", ...
'  '<SyncSeconds>.<SyncNanoseconds>', <UUID1 Offset>, <UUID1
1WayDelay>, <UUID2 Offset>, <UUID2 1WayDelay>
'  ...
' Where:
```

```
'    <SyncSeconds>.<SyncNanoseconds> indicates the sync PTP
' time when the offsets and delays were valid <UUIDx Offset>
' and <UUIDx 1WayDelay> are the Offset from master and One
' way delay as reals in seconds
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Dim m_lastSyncSeconds As PtpManagerC.BigSeconds
Dim m_lastSyncNanoSeconds As Long

Sub Main()
    Dim result As Long
    m_lastSyncSeconds.upperSeconds = 0
    m_lastSyncSeconds.lowerSeconds = 0
    Open "PtpSyncLog.csv" For Output As #1
    result = FindActiveClocks()
    LogSynchronizationHeaders
    result =
PtpManagerC.ptpSetAttribute(PTP_ATTR_MANAGER_CAPTURE_HISTORY
, 1)
    If (result <> 0) Then
        MsgBox "Error Setting Capture History Attribute"
        Exit Sub
    End If

    result = PtpManagerC.ptpInstallSyncMsgHandler(AddressOf
MessageReceivedHandler)
    If (result <> 0) Then
        MsgBox "Error installing message received handler."
        Exit Sub
    End If

   MsgBox "Capturing sync history to 'PtpSyncLog.csv'. Click
'Ok' to stop capture."
```

```
    result =
PtpManagerC.ptpSetAttribute(PTP_ATTR_MANAGER_CAPTURE_HISTORY
, 0)

    result =
PtpManagerC.ptpUninstallSyncMsgHandler(AddressOf
MessageReceivedHandler)

    Close #1


End Sub


Function MessageReceivedHandler(ByVal receiptSecondsUpper As
Long, ByVal receiptSecondsLower As Long, ByVal
receiptNanoseconds As Long, ByVal msgSource As Long) As Long

    Dim result As Long

    Dim msgType As Byte


    result =
PtpManagerC.ptpGetMsgAttribute(PTP_MSG_MESSAGE_TYPE,
msgType)

    If (result <> 0 Or msgType <> PTP_MSG_MESSAGE_TYPE_SYNC)
Then

        MessageReceivedHandler = result

        Exit Function

    End If


    LogLastSynchronizationInfo


    ' Get the time stamp for this sync interval in
preparation for logging next sync message

    result =
PtpManagerC.ptpGetMsgAttribute(PTP_MSG_ORIGIN_TIMESTAMP_SECO
NDS, m_lastSyncSeconds)

    result =
PtpManagerC.ptpGetMsgAttribute(PTP_MSG_ORIGIN_TIMESTAMP_NANO
SECONDS, m_lastSyncNanoSeconds)


    MessageReceivedHandler = 0

End Function
```

```
Function FindActiveClocks() As Long
    Dim result As Long
    Dim i As Byte
    Dim clockCount As Long
    Dim clkInx As Long

    ' Set to search for clocks in all domains
    For i = 0 To 127
        result = PtpManagerC.ptpAddSearchDomain(i)
    Next i

    ' Get data for all clocks
    result = PtpManagerC.ptpUpdateClocks()

    FindActiveClocks = result

End Function

Sub LogSynchronizationHeaders()
    Dim header1, header2, clockSeparator, clockID As String
    Dim result As Long
    Dim clockCount, clkInx As Long
    Dim domain As Byte

    header1 = "SyncTime"
    header2 = ""
    clockSeparator = ", "

    result = PtpManagerC.ptpGetClockCount(clockCount)

    For clkInx = 0 To clockCount - 1
        Dim clock As Long
```

```
        result = PtpManagerC.ptpGetClock(clkInx, clock)
        If (result = 0) Then
            ' Get the clock ID
            result =
PtpManagerC.ptpGetClockStringAttribute(clock,
PTP_ATTR_DEFAULTDS_CLOCK_IDENTITY, clockID)
           result = PtpManagerC.ptpGetClockAttribute(clock,
PTP_ATTR_DEFAULTDS_DOMAIN_NUMBER, domain)
          header1 = header1 & clockSeparator & clockID & "["
& domain & "]"
            header2 = header2 & ", Offset, Delay"
            clockSeparator = ",, "

        End If
    Next clkInx

    Print #1, header1
    Print #1, header2
End Sub

Sub LogLastSynchronizationInfo()
    Dim result As Long
    Dim syncLine, clockEntry As String

    ' Use a fixed string of reasonable length for the
ptpTimeToString return
    Dim lastSyncTime As String * 256

   Dim clockCount, clkInx, histInx, histCount, clock As Long
    Dim histSec As PtpManagerC.BigSeconds
    Dim histNsec As Long
    Dim clkOffset, clkDelay As Double
    Dim histEntryFound As Boolean

    ' If no previous sync time recorded, no info to gather
```

```
    If (m_lastSyncSeconds.upperSeconds = 0 And
m_lastSyncSeconds.lowerSeconds = 0) Then Exit Sub


    result = PtpManagerC.ptpGetClockCount(clockCount)


    ' Get the PTP time of the last sync as a string (note it
is a fixed string that needs later trimming)
    result = ptpTimeToString(lastSyncTime, 256,
m_lastSyncSeconds.upperSeconds,
m_lastSyncSeconds.lowerSeconds, m_lastSyncNanoSeconds, 0)


    ' Quote the time so Excel won't try to convert it to a
float, losing precision
    syncLine = "'" &
PtpManagerC.TrimFixedString(lastSyncTime) & "'"


    result = PtpManagerC.ptpGetClockCount(clockCount)


    For clkInx = 0 To clockCount - 1
        histEntryFound = False
        result = PtpManagerC.ptpGetClock(clkInx, clock)
        If (result = 0) Then
            result =
PtpManagerC.ptpGetClockHistoryCount(clock, histCount)
            For histInx = 0 To histCount - 1
                result =
PtpManagerC.ptpGetClockHistoryAttribute(clock, histInx,
PTP_ATTR_HISTORY_TIMESTAMP_SECONDS, histSec)
                result =
PtpManagerC.ptpGetClockHistoryAttribute(clock, histInx,
PTP_ATTR_HISTORY_TIMESTAMP_NANOSECONDS, histNsec)


                If (histSec.upperSeconds =
m_lastSyncSeconds.upperSeconds And _
                    histSec.lowerSeconds =
m_lastSyncSeconds.lowerSeconds And _
                    histNsec = m_lastSyncNanoSeconds) Then
```

```
                    result =
PtpManagerC.ptpGetClockHistoryDoubleAttr(clock, histInx,
PTP_ATTR_HISTORY_OFFSET_FROM_MASTER, clkOffset)
                    result =
PtpManagerC.ptpGetClockHistoryDoubleAttr(clock, histInx,
PTP_ATTR_HISTORY_PATH_DELAY, clkDelay)
                    clockEntry = ", " & clkOffset & ", " &
clkDelay
                    syncLine = syncLine & clockEntry
                    histEntryFound = True
                End If
            Next histInx
            If (Not histEntryFound) Then syncLine = syncLine
& ",,"
        End If
    Next clkInx

    Print #1, syncLine

End Sub
```

Where to Go Next

The example programs in this chapter, plus additional example programs, are available as part of the IO Libraries Suite 15 typical installation. Refer to the directory:

```
C:\Program Files\Agilent\IO Libraries Suite\
ProgrammingSamples\
```

For example, we've provided a C language example named: **PtpSyncLogger**. This program finds the active clocks, then logs the synchronization history for those clocks to stdout. The format of the output is a comma-separated-value format suitable for consumption by Microsoft Excel.

Also, refer to the LXI help files:

```
C:\Program Files\Agilent\IO Libraries Suite\
InteractiveLXI.chm
```

```
C:\Program Files\Agilent\IO Libraries Suite\
LxiManagement.chm
```

For information regarding VISA or SICL programming, refer to the VISA and SICL Users Guides (visa.pdf and sicl.pdf) as well as the VISA and SICL online help files.

For additional information on Microsoft C# programming, including related information such as .NET Framework, refer to Microsoft's official site:

```
http://msdn.microsoft.com/msdn
```

3
LXI Events and Triggering

This chapter provides general information about LXI Events and Triggering.

NOTE

For specific information on programming LXI Events and Triggers, refer to your specific product documentation and also the programmers API documentation (Appendix A in this manual).

LXI Events

LXI Events are system occurrences that require some type of action. Typically, either the system control program generates the event(s) or an instrument generates the event(s) as a result of some measurement. Within LXI, there are two general event applications:

Event Time Stamping

Event Timestamps allow post acquisition correlation of data and events based on the timestamps. For example, it is possible to capture a timestamp for a trigger signal for both the measuring and source (actuating) services. Timestamps indicate the time in the past when an event occurred.

Triggering

Often in Test and Measurement (T&M) systems, you may need to capture an event, such as a threshold crossing, in one device and to use this event to trigger another device. For example, you can have one device monitor and detect an application significant event, capture a timestamp for the event, and then broadcast an event message. Devices receiving the event message must be preprogrammed to take some action based on the event and timestamp.

LXI Class A and B devices, which include IEEE 1588 Precision Time Protocol (refer to Chapter 4 in this manual) clock synchronization capabilities, provide for time-based triggers. A time-based trigger is analogous to an alarm clock -- it can be set to “go off” at a predetermined time. Instruments responding to the trigger are preprogrammed to perform some specific function. This allows different instruments in a test system to execute complex sequences of events autonomously, without the intervention of the system controller.

LXI Time Stamping

All LXI Class A and Class B devices must assign a time stamp to all measurement data and all direct LAN message events. These time stamps are derived from the IEEE 1588 time base -- that is they use the system “common sense of time.” Note that if an LXI Class C device generates LAN message events, they must use a time stamp of zero*.

The LXI time stamp is defined as the number of seconds and fractional seconds since January 1, 1970. Time stamps are represented programmatically as two doubles: TimestampSeconds and TimestampFraction. LxiEventTime has corresponding 'Seconds' and 'FractionalSeconds' properties.

* If the timestamp is 0 (zero), it means that the event/trigger is immediate or the sender doesn't know the IEEE 1588 time (an LXI Class C device for example).

LXI Triggering

LXI provides three main trigger mechanisms:

- **LAN-based Triggering.** These can be **Driver Command based** where a driver interface on the system controller computer directly transmits a trigger command to an LXI instrument (required on Class A, B, and C devices) OR **Direct LAN messaging based** where a data packet with trigger information (including a timestamp) is sent directly from one LXI device to another via LAN (required on Class A, and B devices). In general, use LAN Triggering if:
  - Instruments are far apart
  - You need the time stamp to look back at previously stored data
  - You want to eliminate cabling from the test system
- **Time-based Triggering**. From the IEEE 1588 Precision Time Protocol running over the LAN interface. Required on Class A and B devices. In general, use time-based triggers if:
  - You need very low latency (distance doesn't matter)
  - You can schedule events in advance
- **A Hardware (wired) Trigger Interface (LXI Trigger Bus).** Required for Class A devices only. Use hardware triggers if:
  - You need very repeatable timing, low jitter
  - You need low latency, and the distance is short

Within the performance limitations of each trigger mechanism, trigger functions can be performed by any of the methods and can be connected together. For example, a trigger event on the wired trigger bus can initiate a LAN or IEEE 1588 trigger event.

Additionally, all classes of LXI instruments are allowed to support traditional vendor specific hardware triggering methods (BNC, SMB, etc.), without restriction. For example, in a simple data acquisition system, a switch module's Relay Closed output may connect to a DMM's External Trigger Input via a BNC-BNC coaxial cable; when a relay closes, the DMM is triggered to initiate its next measurement. In this example, the DMM was preprogrammed to respond to the trigger input.

The LXI standard permits the same type of hardwired triggering but adds other triggering options. For detailed triggering information, refer to your instrument's documentation. Class A and B devices having an Arm-Trigger state machine provide a minimum of eight LAN event inputs for arm and trigger purposes and eight LAN event outputs for signaling other devices.

LXI LAN-based Triggering

LAN-based triggers provide programmatically triggered events through driver commands from the controller to the LXI device or by message exchange between LXI devices. They emulate traditional hardware triggers but can carry information that hardware triggers cannot such as trigger slope and time stamps based on synchronized system clocks. For example, LAN Event ID LAN0 is the LAN message analogous to the LXI hardware Trigger Bus line LXI0 and includes the slope of the trigger signal and a timestamp of the trigger.

Trigger Slope

All devices transmitting LAN events, whose signal source (the signal causing the event) is one of the LXI Trigger Bus lines, one of the signals from an Arm-trigger state machine, or based on a logical signal within the device, must encode the transition edge or slope of the trigger signal. That is, you can specify either a leading or positive edge trigger or a trailing or negative slope trigger. For detailed information, refer to your instrument's documentation.

Time Stamp

In an LXI trigger, the transmitted data packet must include a time stamp for the originating trigger event. If the time stamp is 0 (zero) then it indicates the event/trigger is 'immediate' and/or the sender doesn't know the IEEE 1588 time. The interpretation of non-zero time stamps is not defined by the LXI Standard but is unique to each instrument. This allows for application-specific interpretation. For example, for LXI events the time stamp is likely the time when the event occurred. As a second example, consider a digitizer that is continuously making measurements. When a time stamped event occurs, the digitizer returns

the measurement corresponding to that time. This can be extended to provide either positive or negative trigger delays and is limited only be the depth of the circular data buffer of the instrument.

LXI Triggers

The LXI specification supplies two different techniques for LAN trigger communication: a UDP-based “broadcast” trigger and a TCP, peer-to-peer trigger.

UDP-based Triggers: UDP-based triggers implement a one-to-many multicast. Receivers of a multicast message must have registered with a router to receive them. The UDP protocol is faster (lower latency) than TCP-based triggers because it does not guarantee delivery. However, most modern LAN switches reliably transmit UDP data as long as they are not too heavily loaded with LAN traffic.

TCP-based Triggers: TCP-based triggers are device-to-device communications. That is, they cannot simultaneously trigger multiple devices. The TCP protocol implements a combination of handshaking and data flow control to ensure reliable communications; as such, they have a higher latency and lower maximum throughput than UDP-based triggers.

LXI Time-based Triggering

LXI Class A and B devices are required to implement at least one time-based trigger mechanism. A time-based trigger is analogous to an alarm clock -- it can be set to “go off” at a predetermined time. Instruments responding to the trigger can be preprogrammed to perform some specific function. This allows different instruments in a test system to execute complex sequences of events autonomously, without the intervention of the system controller.

For example, consider a simple test system comprised of a digitizer and RF transmitter. The digitizer may be programmed to begin a measurement cycle at a specified time (UTC), end it after one second, wait 100 milliseconds, and then begin another measurement. The transmitter may be programmed to change its frequency during the 100 msecond digitizer pause. With the IEEE 1588-based clock

synchronization, these operations can be executed, with near-perfect timing, with no connections between the instruments other than the LAN.

The use of time-based triggers requires the ability to schedule instrument actions in advance. Time-based triggers cannot be used to respond to asynchronous events. For events that can be scheduled to occur at a known time and can tolerate some minimal timing jitter, these triggers offer a very high performance, low latency triggering mechanism.

NOTE

Using time stamps in an LXI Event as a trigger is not defined in the LXI specification. This allows for application-specific interpretation. If the timestamp is 0 (zero), it means that the event/trigger is immediate or the sender doesn't know the IEEE 1588 time (an LXI Class C device for example).

Jitter

Time-based triggers implemented in the device hardware have essentially no latency and their performance is not degraded by increasing distances between devices. The timing accuracy depends solely on the accuracy of the IEEE 1588 clock synchronization. However, on a regular, periodic basis, the IEEE 1588 synchronization makes small adjustments to the system slave clocks. This creates some small uncertainty (jitter) to the precise timing of any time-based trigger.

LXI Hardware-based Triggering

The LXI specification for Class A devices defines a standard set of eight independent programmable hardware trigger lines or channels arranged in a bus configuration. Hardware triggering provides an alternative to, or complements, LAN-based triggering in applications requiring higher precision or lower latency. The Trigger Bus connects up to 16 devices in either a daisy-chain configuration, a star configuration (as illustrated in the figure below), or a combination. It must be terminated at both ends.

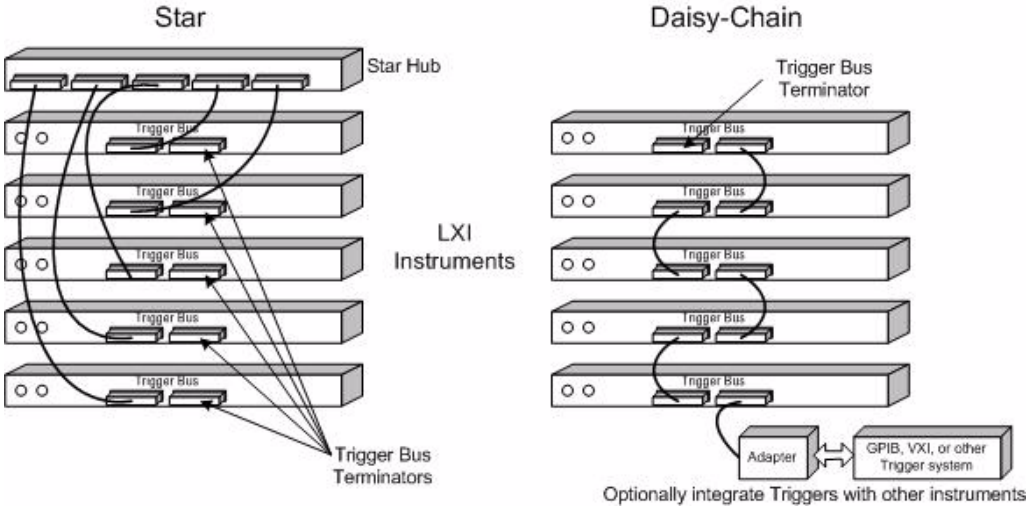


Figure 7 LXI Hardware Trigger Bus

The LXI Trigger bus consist of eight twisted-pair wires. Each channel of the Trigger Bus can operate in one of two user-programmable modes. Note that LXI devices that are not programmed to respond to an impending trigger should have their drivers disabled. The two modes of operation are:

- **Driven Mode:** Provides *single device-to-multiple device* operation. That is, one instrument initiates a trigger to one or more other instruments. A different instrument may simultaneously drive a different trigger channel. As with LAN triggers, each instrument's response to an LXI trigger bus signal is programmable. Likewise, an instrument may use a channel on the trigger bus to transmit an internal signal, since the function of any one trigger channel is not defined.

- **Wired-Or Mode:** This is *multiple device-to-multiple device* operation. That is, one or more instruments can initiate the trigger event to one or more receiving instruments. In this mode, the trigger event can be initiated by the first device to recognize the event OR by the last device ready. The Wired-Or mode requires that one instrument is configured as the Wired-Or Bias device. The bias on the line remains constant until all devices stop driving the bus. This mode of operation is useful whenever the system must wait until several instruments are ready to act.

The Wired Trigger Bus Transmission Line

Figure 8 shows the LXI Trigger Bus connection scheme. Each LXI device has two interchangeable Trigger Bus connectors. The transmission line itself is a twisted pair with a 110Ω (approximate) impedance. At each end of the transmission line is a differential termination load of 100Ω (connected to AC ground by a 0.01µF capacitor).

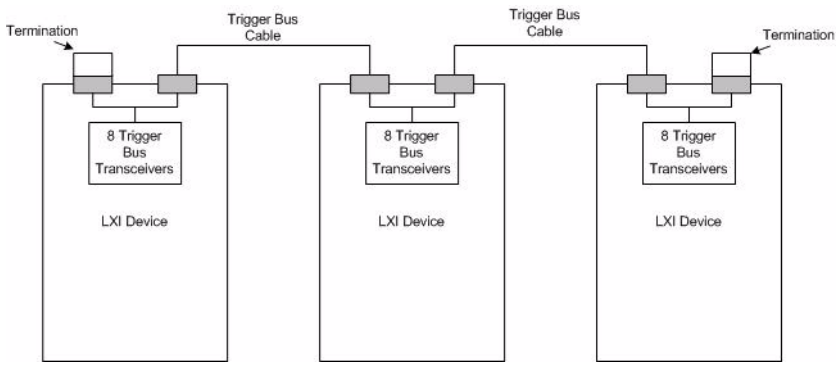


Figure 8 Wired Trigger Bus

Since LXI is a separate instrument standard (as opposed to a card-cage based solution such as VXI), the hardware trigger mechanism is based on the Multipoint Low-Voltage Differential Signaling standard (M-LVDS or just LVDS). LVDS signaling can be carried over long distances and controlled rise/fall time. Refer to Figure 9 for more information on the transmission line and terminators.

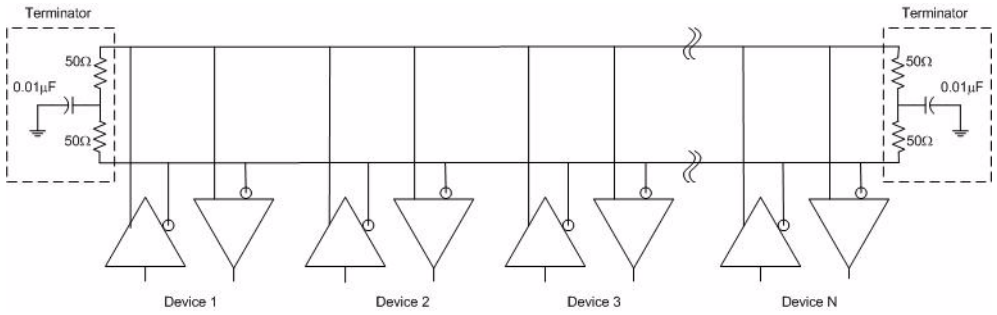


Figure 9 Wired Trigger Bus Transmission Line

Driving the Wired Trigger Bus

As mentioned earlier, the trigger bus has two modes of operation: Driven Mode and Wired-OR Mode. Referring to Figure 9, any one device may assert the logical state of the LVDS trigger bus as long as no other device is simultaneously attempting to drive it. This is known as the Driven Mode - a single device is driving the trigger line to its low or high state. This mode is suitable for a single device-to-multiple device trigger connection.

The LVDS drivers are actually current source drivers rather than voltage source drivers. This allows the trigger bus to function in a Wired-OR mode. In this mode, one device “biases” the trigger line low with one unit of current; all the other devices are disabled (tri-state). The first device to turn its driver from disabled to enabled (trigger event) overcomes the bias device and forces the trigger line high with two units of current. Another way to describe this is when all of the LXI devices on one trigger line are disabled, the net current in the line is negative by one driver’s current (the bias current). The trigger occurs when one device drives two units of positive current onto the line (the net current is one unit positive).

One variation of this mode is called Reverse Operation or Last Device Ready. In this mode, all devices are set to be on with two units of current. As each device is programmed for the trigger event, it sets its driver to the disabled state. The last device to go tri-state allows the bias device to force the trigger bus low initiating the trigger.

4
LXI and IEEE 1588

This chapter provides general information about the IEEE 1588 Precision Time Protocol (PTP) and how it is used in LXI devices. The IEEE 1588 PTP is available on LXI Class A and Class B devices and provides a way to synchronize clocks on these devices providing a common sense of system time. This chapter begins with a general discussion of time, time protocols, and time services and then moves to a more detailed description of the IEEE 1588 PTP and its use in LXI.

NOTE

There are two different IEEE 1588 standards: IEEE 1588-2002 and IEEE 1588-2007. Both standards document the 1588 PTP, but there are distinct differences. For example, the 2002 specification defines PTP Subdomain names as strings but the 2007 specification defines them as integers (0 to 255). Consequently, one instrument designed to the 2002 specification may not work with other instruments designed to the 2007 specification. The 2007 standard also defines new clock types such as End-to-End and Peer-to-Peer Transparent Clocks. Some boundary clock switches designed for the 2007 standard may allow connection between both types of clocks. **Unless otherwise noted, information in this chapter conforms to the IEEE 1588-2007 specifications.**

NOTE

For specific information on using or programming the LXI IEEE 1588 clocks, refer to your specific product documentation and also the programmers API documentation (Appendix A in this manual).

Time in Test and Measurement Systems

Time is a key variable in test and measurement systems. For many instruments (oscilloscopes, logic analyzers, frequency counters, spectrum analyzers, signal sources, etc.), time is a measured or controlled variable. However, these devices treat time individually, as an internal matter. They do not share a system-wide "common sense of time."

In data acquisition systems, precise timing or synchronization can be problematic. This is especially true where the number of devices is quite large. You, the system programmer, must either implement time correlations between multiple sensor readings based on the time of request or receipt of the data at the controller or provide a separate time distribution scheme to permit time stamps to be generated at the source.

For most T&M applications, only relative time within the system is important and no coordination with civil time is necessary. In some applications however, the time base must be traceable to a standard such as UTC. One straight forward way to synchronize the 1588 time base to UTC is to connect the system Grand Master clock to a recognizable UTC source such as GPS or an NTP time server.

Time Scales and Protocols

While there are many civil time systems (such as Loran-C and Network Time Protocol [NTP]), the following five systems are the primary time systems you should be familiar with.

- Local Time is the date/time reported by your PC.
- Temps Atomic International (TAI, French for International Atomic Time) measures real time and is expressed with the date (YYYY-MM-DD) and the time of day (hh:mm:ss). TAI time has been measured continuously since 1955 and is based on cesium beam clocks. As of 01 January 2009, TAI will be ahead of UTC by 34 (leap) seconds. TAI days use the standard Gregorian calendar.

- Coordinated Universal Time (UTC) is also known as Greenwich Mean Time (GMT). UTC is also expressed in the YYYY-MM-DD hh:mm:ss format. Local time differs from UTC by the number of hours of your time zone (e.g., U.S. Mountain Time Zone is UTC minus 7 hours). UTC is based on TAI and very similar except that UTC accounts for Leap Seconds. The difference between UTC and TAI is always an integral number of seconds;
- Global Positioning System (GPS) is the time scale implemented by the atomic clocks in the GPS system (both ground control stations and satellites). GPS time is measured in seconds since the GPS epoch was 0 (zero) at 0 hours, 06 January, 1980. Since it is not changed by leap seconds, GPS time is now ahead of UTC by 14 seconds.
- IEEE 1588 Standard Precision Time Protocol (PTP) is a comprehensive solution for precise time synchronization of multiple instruments containing IEEE 1588 clocks. The time scale is a count of seconds since 01 January, 1970 @ 00:00:00 TAI (it's epoch).

The following chart shows the relative time differences for Tuesday, 24 April, 2007, at about 12:48 p.m. Mountain Standard Time:

Local	2007-04-24 12:48:01	Tuesday	Day 114	Mtn. Timezone UTC-6 hrs.
TAI	2007-04-24 18:48:34	Tuesday	Day 114	UTC plus 33 Leap Seconds
UTC	2007-04-24 18:48:01	Tuesday	Day 311	MJD[1] 54214.78334
GPS	2007-04-24 18:48:15	Week 1424	240495s	Cycle 1 Week 0400 Day 2
IEEE 1588	861475695.000000000			

1 Modified Julian Date

Time Scale Conversion. It is beyond the scope of this manual to describe conversions from all different time scales; however, the following information may help in determining IEEE 1588 time. The PTPTime and LXIEventTime classes provide methods to convert between IEEE 1588 and UTC time and account for leap seconds. The following formulas for converting between PTP time and GPS time was taken from Annex B of the IEEE 1588-2007 Standard:

To convert from	To	Use this formula
GPS Seconds = GPS Weeks x 604800 + GPS secondsInLastWeek	PTP seconds	PTP seconds = GPS seconds + 315964819
PTP seconds	GPS seconds	GPS seconds = PTP seconds - 315964819

NTP Time Service

NTP stands for Network Time Protocol, and it is an Internet protocol used to synchronize the clocks of computers to a UTC time reference. NTP is an Internet standard protocol originally developed by Professor David L. Mills at the University of Delaware.

NTP uses a hierarchical system of “clock strata”; refer to the following figure. Stratum 0 devices, such as GPS clock or other radio clocks, do not connect directly to the network; instead they connect directly to Stratum 1 servers (e.g. via an RS-232 connection using a Pulse per second signal). Stratum 1 servers, also called timeservers, act as servers for timing requests from Stratum 2 servers via NTP. Stratum 2 servers sync to Stratum 1 servers and act as servers for Stratum 3 NTP requests. Stratum 3 (and higher up to 16 levels) servers provide NTP for higher strata.

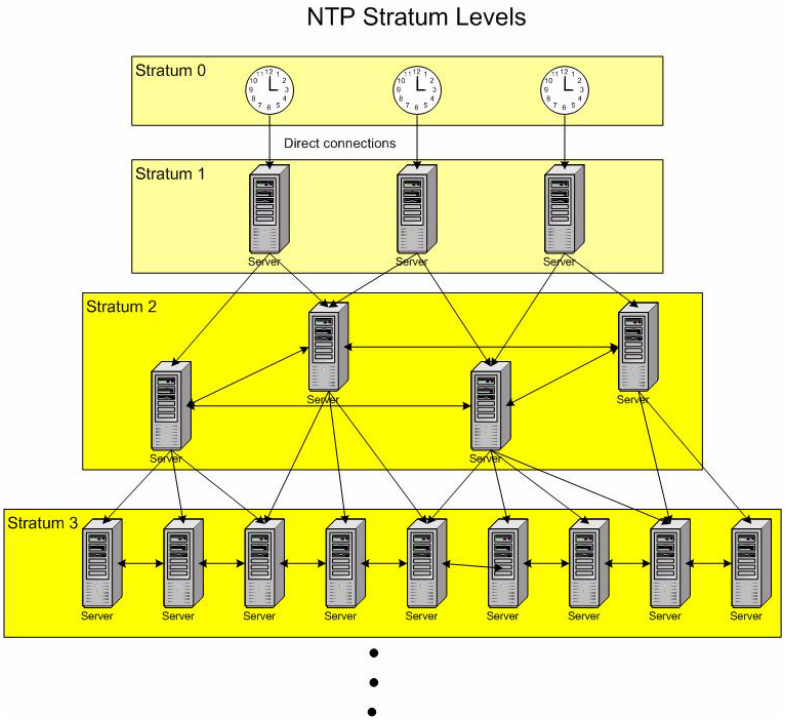


Figure 10 Stratum Levels

For most T&M applications, the NTP time service provides adequate synchronization to the civil UTC time standard but may be a few micro- or even milliseconds different. You corporate IT department will have specific information about synchronizing your test system PC with an NTP time server. For additional NTP information refer to:

- NTP Home Page:
  http://www.ntp.org

- A Microsoft source for NTP information:
  http://support.microsoft.com/default.aspx?scid=kb;EN-US;q262680

Leap Seconds

UTC is occasionally adjusted by one second increments to ensure that the difference between a uniform time scale defined by atomic clocks does not differ from the Earth's rotational time by more than 0.9 seconds. The leap second system was introduced at the beginning of 1972. At that point, UTC equaled TAI minus 10 seconds. This leap second correction can be either positive or negative depending on the Earth's rotation. Since the first leap second in 1972, all leap seconds have been positive and there were 24 leap seconds in the 34 years to January, 2009. This pattern reflects the general slowing trend of the Earth due to tidal braking.

Leap seconds occur only at the end of a UTC month, and have only ever been inserted at the end of June 30 or December 31. Historically, leap seconds have been inserted about every 18 months. However, the Earth's rotation rate is unpredictable in the long term, so it is not possible to predict the need for them more than six months in advance. The sequence of events is:

TAI and GPS Time (no Leap Seconds)	UTC Time (with Leap Seconds)	
23h 59m 57s	23h 59m 57s	
23h 59m 58s	23h 59m 58s	
23h 59m 59s	23h 59m 59s	
00h 00m 00s	23h 59m 60	Leap Second occurred
00h 00m 01s	00h 00m 00	
00h 00m 02s	00h 00m 01	
	00h 00m 02s	Notice that the time is now different by 1 second.

NOTE 31-December-2008 is the most recent leap second addition.

Be aware that the leap second corrections can result in missing time values in UTC but not in IEEE 1588. Thus computing the interval between two time stamps from an IEEE 1588 time base will give the correct answer whereas the difference between two UTC time stamps must involve correction for any leap second insertion in the interval. On the other hand specifying a time in calendar time, for example every day at noon, is easy in UTC but may require a leap second correction to obtain the correct time in an IEEE 1588 time base.

The current number of leap seconds is represented in PTP by the value of **currentUTCOffset**.

Operating Systems Windows XP and Windows VISTA systems that are part of a domain running NTP will automatically synchronize their clock to the domain controller. However, if your system is not part of a domain then the PC clock will drift significantly over time.

Some operating system clocks are aware of leap seconds, others are not. If an operating system is aware of leap seconds then NTP can pass the leap second announcement to the kernel and it's up to the kernel to compensate. Some operating systems, such as Windows, do not care about the leap second announcement, so the 1 second offset after the leap second has been inserted is compensated by stepping the clock back. This may happen with some delay, depending on the NTP polling interval and implementation of the operating system clock.

Calculating Time Spans

Suppose that you have two date/times T1 and T2, with T1 earlier than T2. In general it is easy to subtract T1 from T2 to get the time interval or time span. For example, if T1 = 2006-11-07 13:17:16 and T2 = 2006-11-07 13:17:23, then the time span is seven (7) seconds.

However, what if a leap second was inserted between T1 and T2? If you simply subtract T1 from T2, does the difference include the leap second? For example, suppose you were studying and analyzing data

taken in 2005, and an event occurred at 23:59:53 UT on December 31, followed by a second event at 00:00:10 UT the next day (01-January-2006). You'd think that the delay time between the two would be 17 seconds, but it's really 18 seconds because of the leap second introduced that day. That's a vital difference for the data analysis. Similar problems can occur with the change in daylight savings time (especially with the recent changes in start and end dates), leap years, etc.

Leap second corrections can result in missing time values in UTC but not in PTP (PTP time is the number of *seconds.nanoseconds* since its epoch). Thus computing the interval or time span between two time stamps from a PTP time base will give the correct answer whereas the difference between two UTC timestamps must involve correction for any leap second correction in the interval. On the other hand specifying a time in calendar time, for example every day at noon, is easy in UTC but may require a leap second correction to obtain the correct time in a PTP time base.

NOTE

The timescale actually represented in a PTP subdomain depends entirely on the time scale implemented by the Grandmaster Clock. PTP itself tracks only elapsed time since the epoch. Any correction of PTP times for leap seconds in converting to or from UTC is the responsibility of the application.

More Information

Here's a link to a NIST white paper on enabling NTP clock synchronization with NIST time servers on Windows 2000 and XP:

http://tf.nist.gov/timefreq/service/pdf/win2000xp.pdf

Here's a link to a Microsoft white paper on Windows Time Service:

http://technet2.microsoft.com/WindowsServer/en/library/71e76587-28f4-4272-a3d7-7f44ca50c0181033.mspx?mfr=true

The default behavior of the Windows Time Service on WinXP Pro and Win2K is to get the time from a neighboring PC in a domain hierarchy for the domain the PC finds itself in, with a resulting synchronization within 2 seconds for PC's in the domain (this can

become as much as 20 seconds in large domains). Switching to synchronization to a NIST time server improves that accuracy to between 1 and 50 mS.

- Here's a link to a US Naval Observatory white paper on leap seconds:

  http://tycho.usno.navy.mil/leapsec.html

  This contains a sub-link to a table of exactly when leap seconds have been added to get to the current total of 33 leap seconds for adjusting to atomic time. The most recent leap second addition will be on 01 January, 2009. However, the table shows a quite irregular pattern for when leap seconds have been added. The minimum time between leap seconds additions is about six months (since 1972, when the current system for calculating them was instituted). The maximum time is about six years. Text Bulletin C from the IERS (International Earth Rotation and Reference Systems Service) standards body is mailed every six months to announce a new leap second or confirm that the next June/December month-end does not require one.

IEEE 1588 Precision Time Protocol

The IEEE 1588 Precision Time Protocol (PTP) is a distributed, networked system consisting of a combination of PTP and non-PTP devices. PTP devices include ordinary clocks, boundary clocks, transparent clocks, and management nodes. Non-PTP devices include ordinary network switches, routers and other LAN infrastructure devices, and typically computers, printers, and other application devices.

The PTP protocol is a distributed protocol specifying how the real-time PTP clocks in the system synchronize with each other. These clocks are organized into a master-slave hierarchy with the clock at the top of the hierarchy - the grandmaster clock - determining the reference time for the entire system. Clocks synchronize by exchanging PTP timing messages, with the slaves using the timing information to adjust their clocks to the time of their master in the hierarchy.

The PTP protocol executes within a logical cluster of devices called a PTP domain. Unless otherwise specified, all PTP messages, data sets, state machines and all other PTP entities are always associated with a particular PTP domain. A given physical network and individual devices connected to the network can be associated with multiple PTP domains. Within this standard the time established within one PTP domain by the PTP protocol is independent of the time in other PTP domains.

LXI and IEEE 1588

The LXI standard requires that all Class A and B devices establish a subdomain-wide “common sense of time” based on the IEEE 1588 Precision Time Protocol (PTP). Figure 11 shows a simple Test and Measurement system domain.

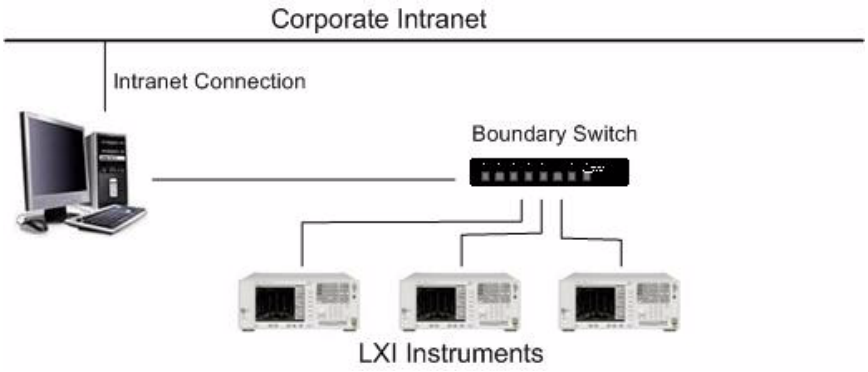


Figure 11 Simple Test and Measurement System

Each LXI Class A and Class B must comply with IEEE 1588 PTP functionality. Specifically, each LXI Class A and B device must be capable of IEEE 1588 Master Clock functionality and the LXI specification recommends that they implement time to a precision of 40 nanoseconds or better. The specification also recommends that the master timebase of all LXI systems is traceable to a UTC timebase (TAI, GPS, NTP, etc. are acceptable). The current PTP time is always defined as the number of *seconds.nanoseconds* since 01-January-1970 (the PTP epoch). From a system view, the precision, accuracy, and epoch of this common sense of time are important:

Precision

There are two facets to the precision of the common sense of time: the precision of an individual device's clock and the collected precision of the networked system. The precision of an individual device's clock is a function of the resolution and characteristics of the clock (oscillator) within the module. The collected precision of the system depends on the precision of the individual devices as well as the precision that IEEE 1588 is able to achieve in synchronizing the clocks within the subdomain.

Accuracy

The accuracy of the common sense of time is a measure of the agreement between the system Master Clock and a time standard such as UTC. In Figure 11, the PC connects to an NTP time server through the corporate intranet; this is the Grandmaster clock and the PC becomes the system or domain Master Clock. Since IEEE 1588 uses a master/slave synchronization protocol, the system accuracy is determined by the accuracy of the Grand Master Clock time base. Each Slave Clock adjusts its time base to the agree with its Master Clock and will have the same accuracy.

NOTE

The term ‘Grandmaster’ defines the “root” master of all clocks in a system of IEEE 1588 clocks. Since IEEE 1588 partitions clocks into a tree hierarchy of master/slave clocks, there may be multiple Master Clocks, but only one will be the Grandmaster Clock at the root of the tree.

Epoch

A time base epoch is the origin of the time as measured in the system. In IEEE 1588, the epoch is 01-January-1970 00:00:00 TAI. The epoch field in PTP time has been merged into the seconds field as its most significant 16 bits.

IEEE 1588 Time Base

IEEE 1588 time bases are continuous from their epoch. If a UTC time base is required then the issue of leap seconds must be managed. IEEE 1588 time base's do not insert leap seconds (unlike NTP), but they do provide information on when they will occur and how may have occurred to date. The international UTC time standard does insert leap seconds, with a 60th second at the end of June or December when the international time authorities decide one is needed to align UTC with astronomical time.

The 'UTC' time that MS Windows exposes is NOT international UTC. Rather, it is intended to align with the local time at Greenwich (Greenwich mean time) and does NOT account for leap seconds. That means that immediately after a leap second occurs, the UTC-windows time will be a second ahead.

Note that the leap second corrections can result in missing time values in UTC but not in IEEE 1588. Thus computing the interval between two time stamps from an IEEE 1588 time base will always give the correct answer whereas the difference between two UTC time stamps must involve correction for any leap second insertion in the interval. On the other hand specifying a time in calendar time, for example every day at noon, is easy in UTC but may require a leap second correction to obtain the correct time in an IEEE 1588 time base.

No distinction is made in the IEEE 1588 protocol between a software clock and a hardware clock. However, to be able to work with synchronization in the nanosecond range, hardware support is required. Generally, synchronization errors caused by software jitter cannot be eliminated. With a pure software solution (e.g. Windows systems), the error may actually be in the micro- or milli-second range.

IEEE 1588 Clock Synchronization

The basic process for IEEE 1588 clock synchronization involves a two-step process:

- Establishing the Master-Slave hierarchy
- Synchronizing the clocks

IEEE 1588 PTP is a Master/Slave process; the most precise clock on the network, known as the Master Clock, synchronizes all the other Slave Clocks in the subdomain. By the 1588 protocol, clocks are categorized by their precision into various classes (stratum). The selection of the best clock on the network is accomplished automatically with a "Best Master Clock" algorithm.

Best Clock Algorithm

Also known as the Best Master Clock Algorithm, the IEEE 1588 protocol provides a process for determining which clock in the system is the best clock and making that the Master Clock for the local network. The algorithm consists of two parts: a *State Decision Algorithm* and a *Data Set Comparison Algorithm*

State Decision Algorithm This algorithm computes the recommended state (Master, Slave, or Passive) for each port of each clock and for each clock as a whole. This state is based on the Data Set Comparison Algorithm. For example, if a given clock finds that the best of the best data sets received from all its ports is better than itself, then it becomes a slave to the best clock on that port. After each clock goes through this algorithm, the data set for each clock is updated.

Data Set Comparison Algorithm IEEE 1588 maintains state information, i.e. Data Sets, for each clock and for each port of each clock — four Data Sets for the clock as a whole and two Data Sets for each port of the clock. Data Sets are defined in the IEEE 1588-2007 standard. The algorithm is based on binary comparisons of attributes with the following precedence:

1. Priority 1: If one clock is the user-specified "preferred" clock, then that clock is chosen.
2. Clock Class: An attribute defining a clock's TAI traceability.
3. Clock Accuracy: An attribute defining the accuracy of the clock.
4. PTP Variance: An attribute defining the stability of the clock.
5. Priority 2: A user-configurable designation ordering otherwise equivalent clocks.
6. Clock Identity: A tiebreaker based on unique identifiers.

For detailed information on the algorithms, data sets, clock identifiers, clock variances, etc., refer to the IEEE 1588-2007 standard.

Synchronizing Clocks

Each Slave Clock synchronizes to its Master Clock by exchanging four types of synchronizing messages:

- Sync
- Follow_Up
- Delay_Request (or Delay_Req)
- Delay_Response (or Delay_Resp)

This process has two stages. In the first stage, the time difference between the master and slave is corrected. This is also known as the **Offset Measurement** and occurs on a regular basis (default Sync Interval is every 2 seconds). The second stage, known as the **Delay Measurement**, determines the delay or latency between the slave and the master. The Delay Measurement occurs infrequently and at larger time intervals (default is every 4 to 60 seconds). The following diagram illustrates the process.

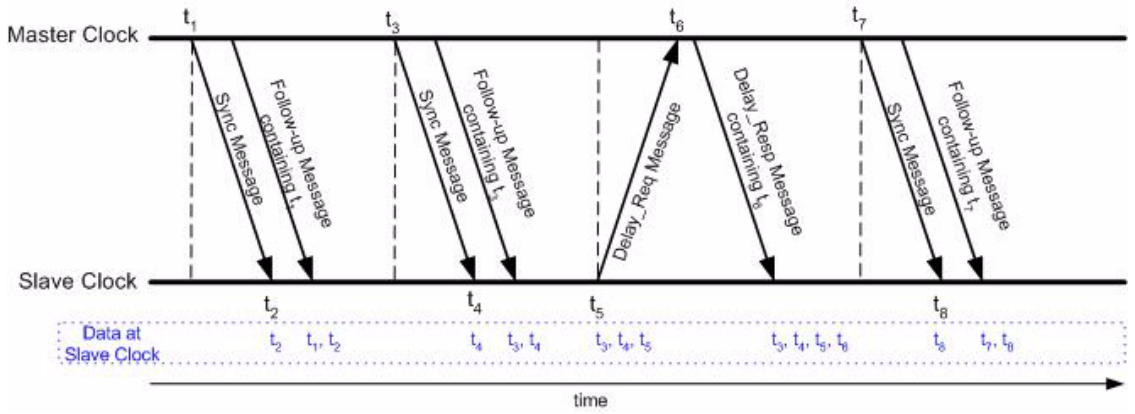


Figure 12 IEEE 1588 Master & Slave Synchronization

At t_1, the Master Clock sends a **Sync** message to a Slave Clock. This **Sync** message contains an estimated value for the time (time t_1) the message was transmitted. The Master Clock monitors the exact time the **Sync** message was transmitted (t_1), and in a second **Follow_Up** message, sends that exact time to the Slave.

Meanwhile, the Slave clock monitors the exact time it received the **Sync** message (t_2). When the Slave Clock receives the **Follow_Up** message, it calculates its correction (*offset* = t_2- t_1 - line_delay) relative to the Master Clock. The Slave Clock then corrects its time by this offset. If there is no delay over the transmission path (i.e., *line_delay* = 0), both clocks are now synchronized. The same process occurs again at times t_3 and t_4 and a new offset is computed.

However, in reality there is a delay over the transmission path; in other words, *line_delay* can never actually be 0 (zero). During the second stage of the synchronization process, the Slave Clock sends a **Delay_Req** message to the Master clock. The Slave monitors the exact time of the transmission (t_5). The Master clock generates a time stamp (t_6) upon receipt of the **Delay_Req** and sends it back to the Slave in a **Delay_Resp** transmission. With this information, the Slave then calculates the delay between the Slave and the Master [*line_delay* = ((t_4 - t_3) + (t_6 - t_5)) / 2] and uses *line_delay* in its offset computation. The process assumes the delay is uniform or symmetrical (the same in both directions) between the Slave and the Master.

What Does it mean to be “In Sync”?

Ultimately each clock, both the Master and Slave, are free running clocks and are subject to drift, jitter, etc. Most Slave clocks use a servo mechanism to “fine tune” the oscillator frequency to better keep synchronized with the Master Clock. Consequently, it may take several synchronizing “cycles” before the Slave Clock approximates the Master Clock. Never-the-less, the Slave clock will never precisely match the Master clock; they will always be some finite number of nanoseconds different. The challenge then is to determine what minimum offset (called Sync Threshold) is acceptable in your test system. Refer to the following figure.

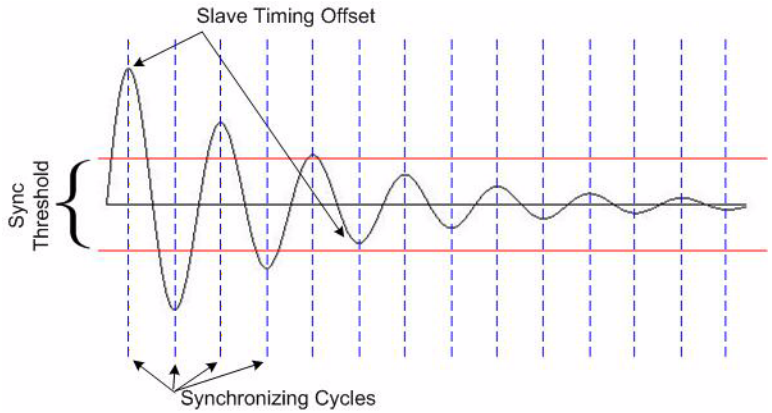


Figure 13 Sync Threshold Example

In this example, it takes six synchronizing cycles before the actual offset becomes smaller than the sync threshold. If the system takes 2 seconds between synchronizing cycles, then it takes a total of twelve (12) seconds for the slave clock to adjust to within the Sync Threshold. Furthermore, you may want to ensure the Slave Clock remains within the Sync Threshold for a number of consecutive synchronizing cycles (Threshold Count) before you consider the Slave to be "synchronized" with the Master clock. If your Threshold Count is 10 cycles, and your system takes 2 seconds between synchronizing cycles, then it takes an additional 20 seconds before your system is considered "in sync" with the Master Clock.

NOTE

Typical values for Sync Intervals range from 1/16 second to 2 seconds (-4 to 1 as a logSyncInterval). It is calculated as the logarithm base 2 of the current sync_interval value; that is, if *sync_value*=1 then the interval is 2^1 =2 seconds.

Jitter and Stability Issues

Jitter

When implemented in the device hardware, time-based triggers have essentially no latency and their performance is not degraded by increasing distances between devices. The timing accuracy depends solely on the accuracy of the IEEE 1588 clock synchronization. However, on a regular, periodic basis, the IEEE 1588 synchronization makes small adjustments to the system slave clocks. This creates some small uncertainty (called jitter) to the precise timing of any time-based trigger.

Stability

The delay fluctuation introduced into the computation of the offset_to_master and one_way_delay variables may be reduced by suitable design of the synchronization servo algorithms in the local instrument clock. Trade-offs must be made between the averaging times (number of samples) and the responsiveness to effects other than delay fluctuation, such as oscillator stability. Likewise a trade-off must be made between sampling rates (sync interval) and responsiveness to changes in topology (selection of Master Clocks) and the required computation and network bandwidth resources.

The fundamental time stability of the local clock must be consistent with the sync interval and accuracy specifications necessary for your test system. The algorithms used to reduce delay fluctuation will not correct for drifts of the local clocks during time intervals small compared with the averaging intervals of the algorithms. Servos cannot correct for random drifts occurring within a sync interval.

At high accuracy the specifications on the stability of the local oscillators driving the local clock can be quite difficult to meet. The trade-off will be between cost and stability. Local oscillators will typically be quartz crystals. Quartz crystals typically drift due to thermal, mechanical, and aging effects. Of these, thermal effects are the most difficult in most applications.

For example, a typical thermal specification for uncompensated crystals is 1 PPM per degree Celsius. A one degree temperature rise over a sync interval of 2 seconds will produce an error of roughly 2 microseconds.

The thermal environment of the crystal would need to be controlled to maintain this level. To achieve accuracy in the tens of nanosecond range would require that some combination of better thermal specifications on the crystal, reduced sync interval, or better thermal management combine to reduce the thermal drift by two orders of magnitude.

IEEE 1588 PTP allows sync intervals to range from 1/16 second to 2 seconds (-4 to 1 as a logSyncInterval), with the corresponding increase in computation and network bandwidth requirements.

Other IEEE 1588 Resources

- National Institute of Standards and Technology, IEEE 1588 Website:

  This site provides the complete IEEE 1588 standard, additional information, and tutorials about the standard:

  http://ieee1588.nist.gov/

- Introduction to IEEE 1588:

  http://ieee1588.nist.gov/intro.htm

The book: *Measurement, Control, and Communication Using IEEE 1588 (Advances in Industrial Control)*, John C. Eidson, Springer Science+Business Media, Springer-Verlag London Ltd. 2006

5 Test System Applications and Design Considerations

A summary of best practices for setting up an LXI-compatible network includes:

- Isolate instruments from general network traffic using a switch. Do not use a hub.
- Use a Boundary Switch as that switch for the best timing synchronization. (sub-nanosecond) Using a general purpose switch instead may degrade synchronization to tens of microseconds for a lightly loaded switch.
- Configure all firewalls so that port 5044 is not blocked.
- Configure all routers so that they allow UDP multicast messages.
- When LXI Class A and B instruments are powered on, it can take a few minutes for the IEEE 1588 clock synchronization to occur. Do not start a test program during that time.
- Regularly check Clock synchronization using the Lxi1588Manager's WaitforSynchronization method.

NOTE

Firewalls. Unfortunately, firewalls cannot distinguish LXI and IEEE 1588 requests for incoming connections (for LXI events and 1588 messages) from other, similar requests. You will need to configure your firewall utility to allow incoming connections on the LXI port (5044) and the 1588 ports (typically 319 and 320).

Network Topology

The following figure shows a simple T&M network topology with one domain:

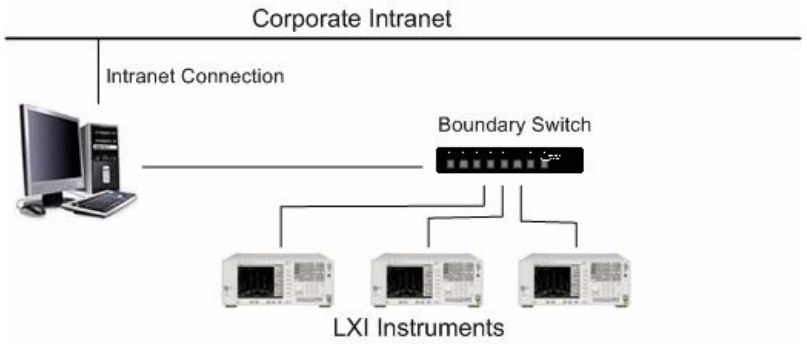


Figure 14 Typical LXI Test and Measurement System

In most LAN systems, all communications between devices goes through a switch or router. These devices could easily limit both the precision and the accuracy of the overall IEEE 1588 time system. The IEEE 1588 solution is to modify these switches to include an IEEE 1588 clock that serves as a time transfer standard. These modified switches are called Boundary Switches (or Boundary Switches) in the IEEE 1588 standard.

Industrial automation and control applications are often very complex, sometimes involving more than 1,000 distributed sensors or actuators and multiple levels of control, each with an IEEE 1588 clock. Figure 15 shows a complex network with several LXI devices, each with an IEEE 1588 clock.

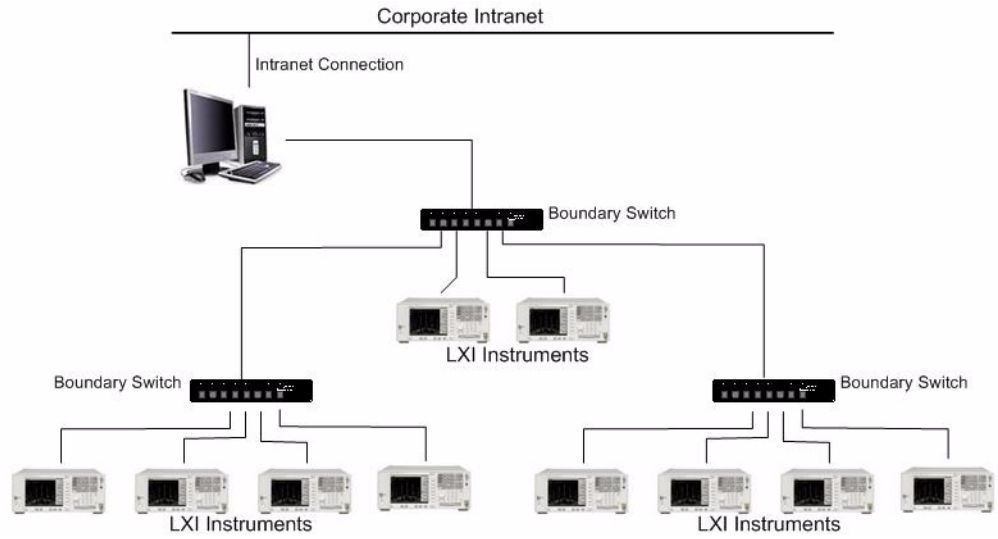


Figure 15 Complex Test and Measurement System with IEEE 1588

The precision of each clock and the IEEE 1588 Boundary Switches and the overall system performance/precision requirements must be considered during system design. Devices requiring tight synchronization with the overall system Master Clock (in this example, the PC) should be located as close as possible to the system Master Clock. Devices that do not require the tight synchronization may be placed further from the system Master Clock.

The instruments in each subdomain in Figure 15 are synchronized to each other. But because of the distance from the system Master Clock, the clocks in the lower two subdomains (bottom row of instruments) will not be as precise and have more jitter than the middle subdomain (middle row of two instruments).

PTP Device Types

LXI Class A and B devices must include IEEE 1588 Precision Time Protocol and Event Triggering. This makes it possible to achieve sub-microsecond synchronization of LXI devices located anywhere on a local network. This class of instruments can use time-based triggers and LAN-based triggers.

IEEE 1588 clock synchronization requires one network device to be the Master Clock to which all other clocks are slaved (synchronized). Because the lowest-latency path in any network is likely to be the path between the router/switch and each network device, the best place to put the 1588 Master Clock is in the switch itself. In the IEEE 1588 specification, such a switch is called a Boundary Switch (or Boundary Clock). For example, a standard, low-load commercial off-the-shelf (COTS) switch introduces 375 nS of jitter, with a standard deviation of 49 nS. Using such a switch should allow sub-microsecond synchronization of clocks. A Boundary Switch, by comparison, introduces 120 nS of jitter, with a standard deviation of 15 nS. That's approximately one-third the impact on synchronization.

The IEEE 1588-2007 specification allows for devices such as End-to-End Transparent Switches and Peer-to-Peer Transparent Switches that are not defined in the IEEE 1588-2002 specification.

NOTE In the case of PTP ordinary clocks and Boundary Switches are properly synchronized, the epoch is the epoch of the timescale in use. For PTP Transparent Switches, the epoch is locally defined and not necessarily aligned with the time scale.

Ordinary Clocks

PTP ordinary clocks communicate with other network devices via the logical interfaces based on a single physical port. The Event Interface sends and receives event messages which are timestamped based on the value of the local clock. The General Interface sends and receives general messages. Each Ordinary clock in the network subdomain supports a single copy of the protocol and has a single PTP state. The clock may be (but is not required to be) the Grandmaster clock in the system or it can be a Slave clock.

Figure 16 shows a very simple block diagram of a simple clock.

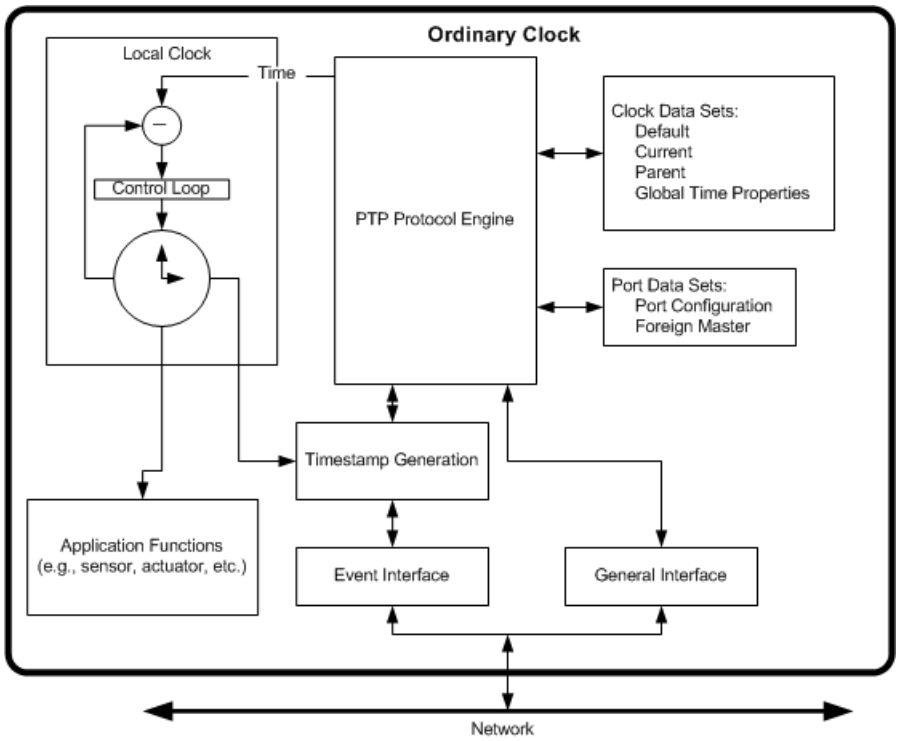


Figure 16 Block Diagram of a PTP Ordinary Clock

Ordinary clocks maintain two types of data sets, referred to as clock data sets and port data sets respectively. The clock data sets are:

- **Default:** Attributes describing the Ordinary clock,
- **Current:** Attributes related to synchronization,
- **Parent:** Attributes describing the clock to which the Ordinary clock synchronizes and the grandmaster.
- **Global time properties:** Attributes of the timescale.

The port data sets are:

- **Port configuration:** Attributes of the port including the PTP state,
- **Foreign master:** Attributes of clocks that are potential masters of this clock.

The protocol engine:

- Sends and receives PTP messages
- Maintains the data sets
- Executes the state machine associated with the port
- Computes the master's time based on the received PTP-timing messages and timestamps that were generated -- If the port is in the slave state (should synchronize to its master).

The control loop in the local clock adjusts the clock to agree with the time of its master if the Ordinary clock port is in the slave state. If the port is in the master state, the local clock is free running or possibly synchronized to an external source of time such as the GPS system. If the port is in the master state and the Ordinary clock is the grandmaster clock of the Domain, then the local clock is typically synchronized to an external source of time traceable to TAI such as the GPS system.

In some applications such as industrial automation, an Ordinary clock will also be associated with an application device such as a sensor or actuator. In a telecommunications application an Ordinary clock may be associated with a timing demarcation device.

Boundary Switches

Boundary Switches are multi-port switches (refer to Figure 17 on page 71) containing one port that is a PTP Slave to a Master Clock, and the remaining ports are Master Clocks to downstream Slave Clocks. They act like transfer standards when a high accuracy Master Clock is in the system. Use of a Boundary Switch can greatly improve the timing accuracy of the IEEE 1588 clock synchronization. The use of a Boundary Switch, while not mandatory, is highly recommended.

Note that Boundary Switches do not pass **Sync**, **Follow-Up**, **Delay_Req**, or **Delay_Resp** messages. Instead, one side of the Boundary Switch acts like a slave to the system Master Clock, and the other side of the Boundary Switch acts like Master Clocks to the other devices on the network.

Each clock on the Boundary Switch is its own port; so in Figure 17, the Boundary Switch has four ports, one for the Master Clock and three for the Slave Clocks. Refer to the following diagram:

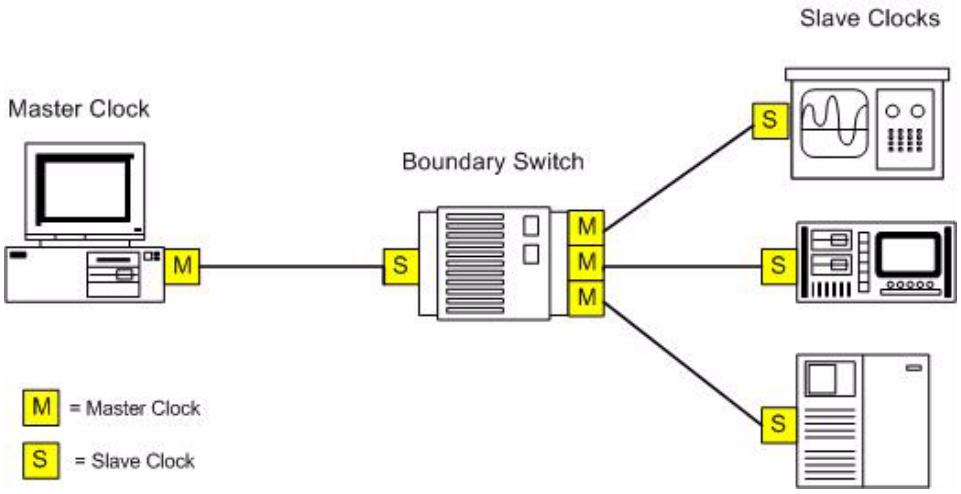


Figure 17 IEEE 1588 Boundary Switch

Figure 18 shows a simple block diagram of a Boundary Switch. Each port of a Boundary Switch is like the port of an Ordinary clock with the following exceptions:

- The clock data sets are now common to all ports of the Boundary clock
- The local clock is common to all ports of the Boundary clock
- Each protocol engine has the additional function of resolving the states of all ports to determine which port provides the time signal used to synchronize the local clock

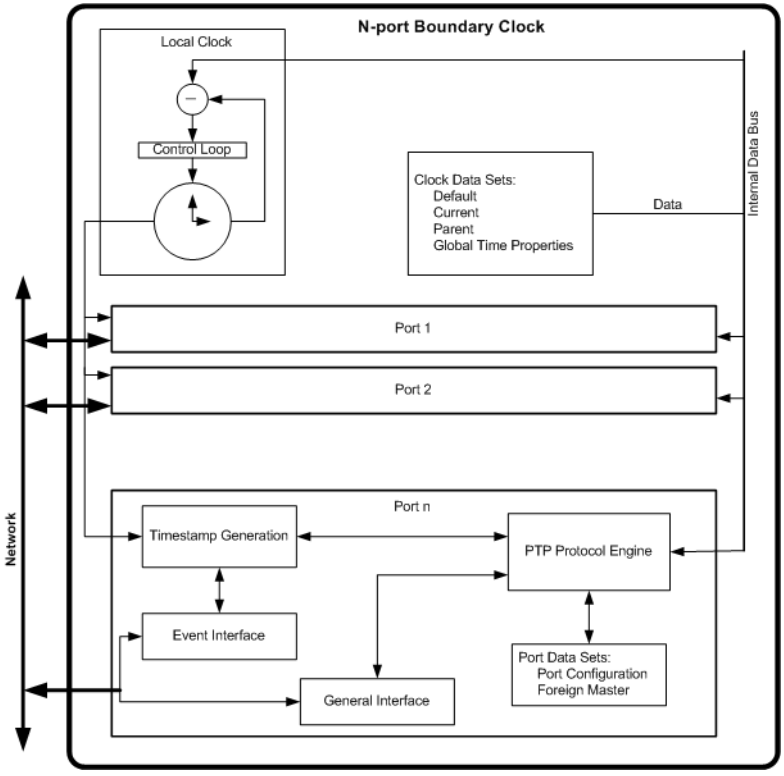


Figure 18 Block Diagram of a Boundary Switch

Messages related to synchronization, establishing the master-slave hierarchy, and signaling terminate in the protocol engine of a Boundary Switch and are not forwarded. Management messages are forwarded to other ports on the Boundary Switch subject to restrictions to limit the propagation of these messages within the system.

This discussion of Boundary Switches applies only to PTP messages. For all non-PTP messages the Boundary Switch behaves as a normal network component, e.g. switch, repeater, or router.

A Boundary Switch is typically used only as a network element and is not normally associated with application devices such as sensors or actuators.

Transparent Switches

Another potential hardware option for PTP-based networks is the Transparent Switch. These switches modify the precise time stamping in the **Delay_resp** and **Follow_up** synchronizing messages (see "IEEE 1588 Clock Synchronization" on page 58) to account for the send/receive latencies within the switch itself. The result is better synchronization between the Master and Slave Clocks. However, Transparent Switches may create security issues when the original packet crypto checksum doesn't match the final packet checksum arriving at the slave.

End-to-End Transparent Switches These switches forward all network messages just like a normal switch, router or repeater. However for PTP-event messages, the residence time bridge measures the residence time of PTP Event messages (the time the message takes to traverse the Transparent Switch). These residence times are accumulated in a special field, the correction field, of the PTP Event message or the associated Follow-up message. This correction is based on the difference in the timestamp generated when the Event message enters and leaves the Transparent Switch. Any updates to checksums required by the network protocol are made. Note that the value of the correction update and checksums are specific to each output port and message since the residence times are not necessarily the same for all paths through the Transparent Switch or for successive messages on the same path.

Timestamps used in computing the residence time are based on timestamps generated from the local clock. Since a slave clock uses the accumulated residence times to adjust the time provided by a master, it is important that any errors resulting from differences in the rates of the master and the Transparent Switches are negligible for the accuracy required by the application. Since it is possible that the rates of the master and local clocks (essentially the definition of the second on the two clocks) can differ by 0.02%, the error introduced is 0.02% of the measured residence time. Thus for a residence time of 1 ms the maximum error is 200 ns. In general these errors will be unacceptably large. To reduce these errors the rate of the local clock can be made equal to that of the master, i.e. synchronized, by observing the timing information in received Sync messages corrected for any upstream

residence times. The residence time bridge performs this function and passes the needed rate corrections to the rate control loop in the local clock.

End-to-End Transparent Switches may be used as network elements or may be associated with application devices such as sensors or actuators.

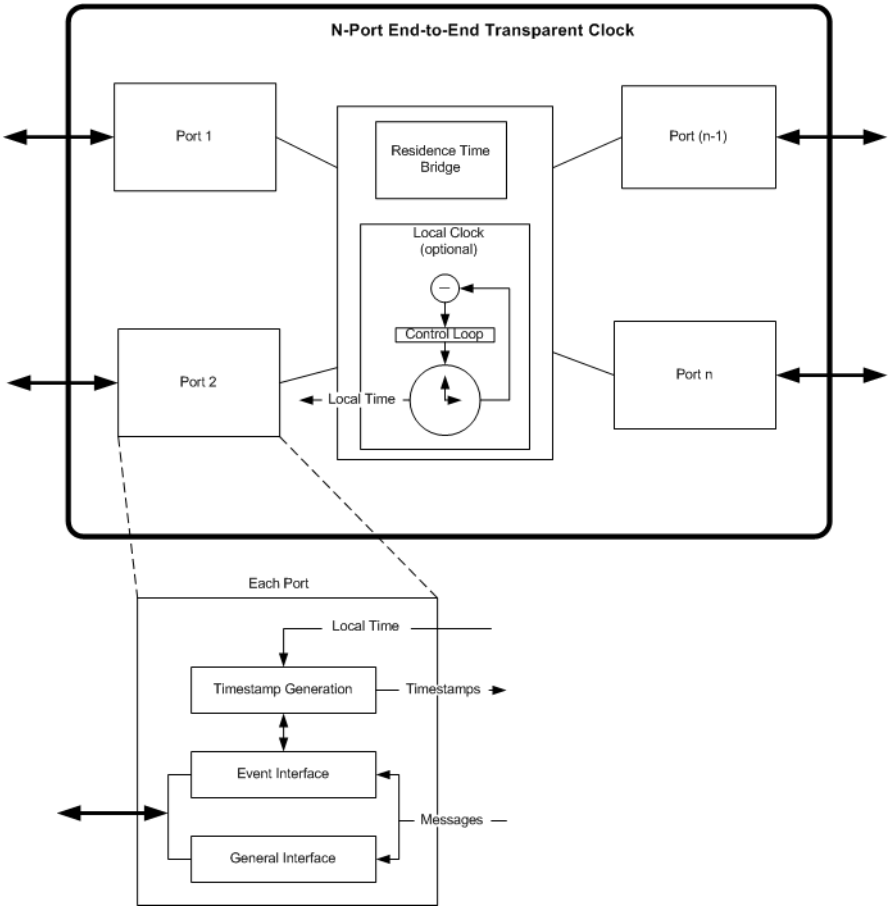


Figure 19 Block Diagram of an End-to-End Transparent Switch

Peer-to-Peer Transparent Switch Peer-to-Peer Transparent Switches differ from the End-to-End Transparent Switches in the way they correct and handle the PTP-timing messages. In all other respects it is identical with the End-to-End Transparent Switch.

The Peer-to-Peer Transparent Switch has an additional per port block to compute the link delay between each port and a similarly equipped port on another node sharing the link -- the link partner. The link partner will be in another clock supporting the peer delay mechanism since non-Peer-to-Peer devices are not expected between Peer-to-peer Transparent Switches. The computation of link delay is based on an exchange of **Pdelay_Req**, **Pdelay_Resp**, and possibly **Pdelay_Resp_Follow_Up** messages with the link partner. As a result of these exchanges the link delay is known for each port of the Peer-to-peer Transparent Switch. Unlike the End-to-End Transparent Switch that corrects and forwards all PTP-timing messages, the Peer-to-peer Transparent Switch only corrects and forwards **Sync** and **Follow_Up** messages. The appropriate correction field in these messages is updated for both the residence time of the **Sync** message within the Peer-to-Peer Transparent Switch and for the link delay on the port receiving the **Sync** message.

NOTE The two Transparent switches do not work together. A Peer-to-Peer Transparent Switch can only work with clock ports supporting the peer delay mechanism. With Peer-to-Peer Transparent Switches, the master only needs to issue Sync and Follow_Up messages and respond to Pdelay_Req messages. It does not receive Delay_Req messages and therefore the processing workload is independent of the number of Peer-to-Peer clocks served by a given port. If a network contains Peer-to-Peer Transparent Clocks in one region and End-to-End Transparent Clocks in another region, the regions can be connected together only through a Boundary Switch.

NOTE LXI does not support use of peer-to-peer transparent switches.

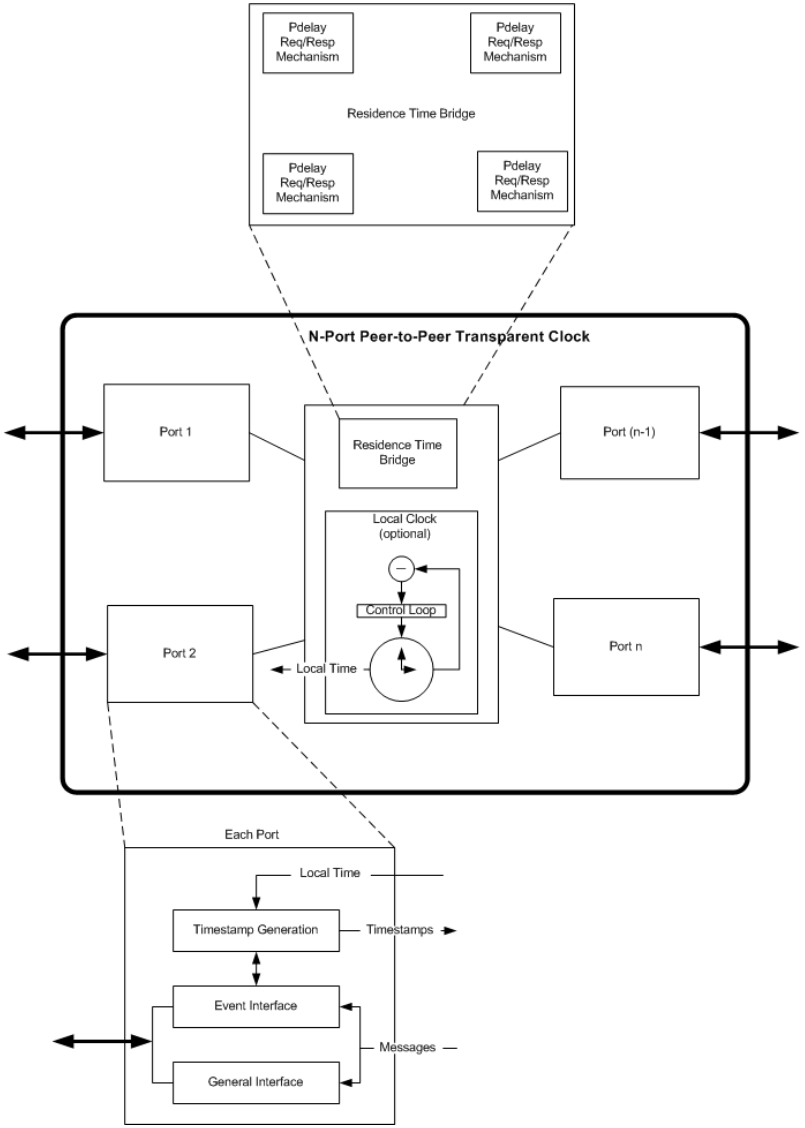


Figure 20 Block Diagram of a Peer-to-Peer Transparent Switch

Appendix A: LXI API Information

This appendix provides information on Agilent LXI API. There are two main parts of the API, one for LXI Timing and the second for LXI Events. For specific information on the API classes and enumerations, refer to the *Lxi Management Help* file, available at the IO Control icon.

LXI EventManager Namespaces

Classes

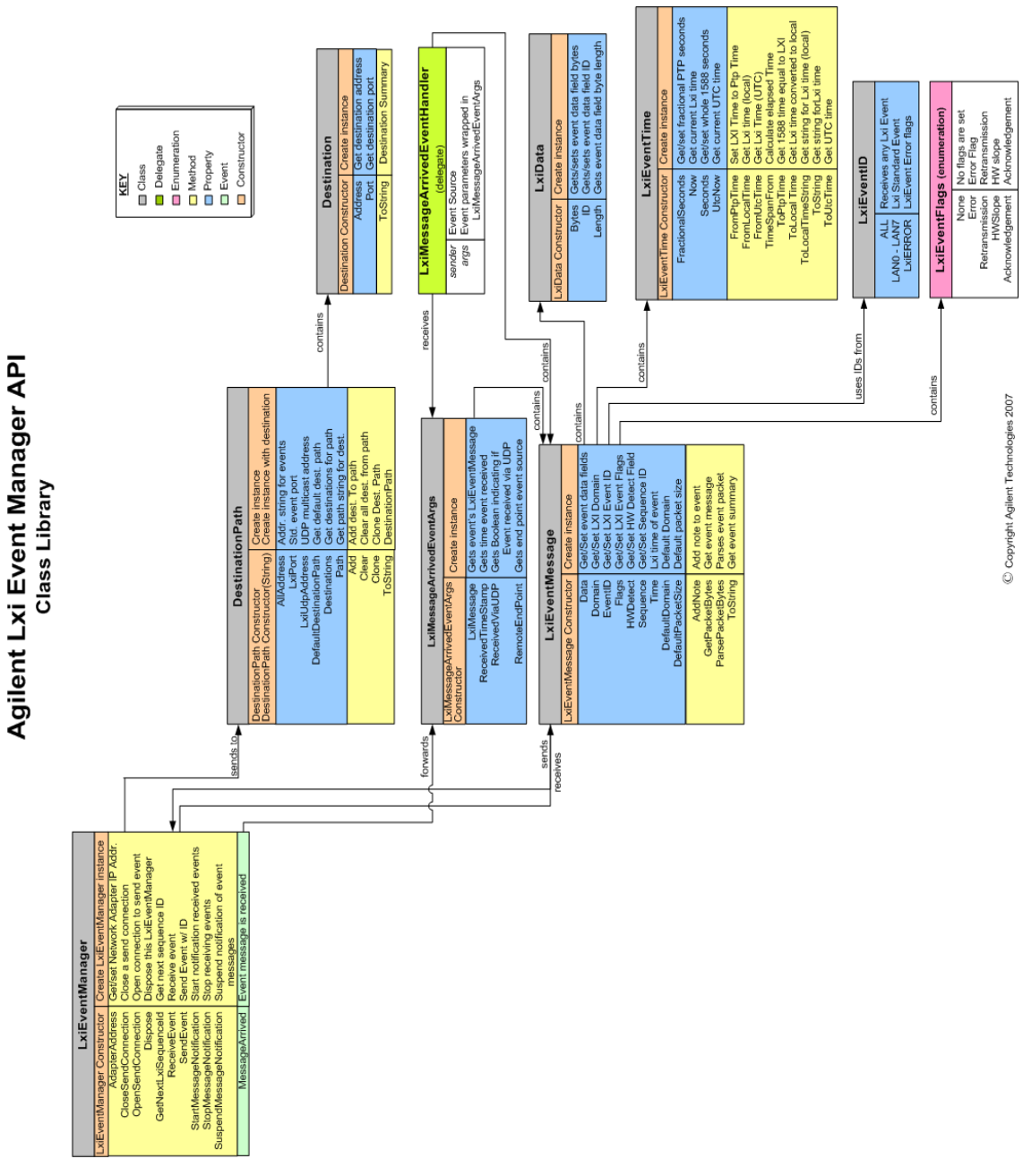
Destination	A destination to receive LXI Events.
DestinationPath	Holds destinations to receive LXI Events.
LxiData	Contains the data associated with an LXI Event data field
LxiEventId	This class provides a list of standard LXI Event ID strings.
LxiEventManager	The manager for sending and receiving LXI events.
LxiEventMessage	Holds LXI event packet information.
LxiEventTime	LXI Event Time representation
LxiEventViewControl	A control for interactive LXI Event send and receive.
LxiMessageArrivedEventArgs	Event arguments for LxiMessageArrivedEventHandler
LxiSendOptionsDialog	LxiSendOptionsDialog allows setting advanced LXI event data fields.

Delegates

LxiMessageArrivedEventHandler	Represents the method that will handle the MessageArrived event.

Enumerations

LxiEventFlags	LXI Event Flags


Agilent Lxi Event Manager API
Class Library
KEY
Class
Delegate
Enumeration
Method
Property
Event
Constructor
LxiEventManager
LxiEventManager Constructor
AdapterAddress
CloseSendConnection
OpenSendConnection
Dispose
GetNextLxiSequenceId
ReceiveEvent
SendEvent
StartMessageNotification
StopMessageNotification
SuspendMessageNotification
MessageArrived
Create LxiEventManager instance
Get/set Network Adapter IP Addr.
Close send connection
Open connection to send event
Dispose this LxiEventManager
Get next sequence ID
Receive event
Send Event w/ ID
Start notification received events
Stop receiving events
Suspend notification of event messages
Event message is received
sends to
DestinationPath
DestinationPath Constructor
DestinationPath Constructor(String)
AllAddress
LxiPort
LxiUdpAddress
DefaultDestinationPath
Destinations
Path
Add
Clear
Clone
ToString
Create instance
Create instance with destination
Addr. string for events
Std. event port
UDP multicast address
Get default dest. path
Get destinations for path
Get path string for dest.
Add dest. To path
Clear all dests. from path
Clone Dest. Path
DestinationPath
contains
Destination
Destination Constructor
Address
Port
ToString
Create instance
Get destination address
Get destination port
Destination Summary
forwards
LxiMessageArrivedEventArgs
LxiMessageArrivedEventArgs Constructor
LxiMessage
ReceivedTimeStamp
ReceivedViaUDP
RemoteEndPoint
Create instance
Gets event's LxiEventMessage
Gets time event received
Gets Boolean indicating if Event received via UDP
Gets end point event source
receives
LxiMessageArrivedEventHandler
(delegate)
sender
args
Event Source
Event parameters wrapped in LxiMessageArrivedEventArgs
sends
receives
contains
LxiEventMessage
LxiEventMessage Constructor
Data
Domain
EventID
Flags
HWDetect
Sequence
Time
DefaultDomain
DefaultPacketSize
AddNote
GetPacketBytes
ParsePacketBytes
ToString
Create instance
Get/Set event data fields
Get/Set LXI Domain
Get/Set LXI Event ID
Get/Set LXI Event Flags
Get/Set HW Detect Field
Get/Set Sequence ID
Lxi time of event
Default Domain
Default packet size
Add note to event
Get event message
Parses event packet
Get event summary
LxiData
LxiData Constructor
Bytes
ID
Length
Create instance
Gets/sets event data field bytes
Gets/sets event data field ID
Gets event data field byte length
LxiEventTime
LxiEventTime Constructor
FractionalSeconds
Now
Seconds
UtcNow
FromPtpTime
FromLocalTime
FromUtcTime
TimeSpanFrom
ToPtpTime
ToLocal Time
ToLocalTimeString
ToString
ToUtcTime
Create instance
Get/set fractional PTP seconds
Get current Lxi time
Get/set whole 1588 seconds
Get current UTC time
Set LXI Time to Ptp Time
Get Lxi time (local)
Get Lxi Time (UTC)
Calculate elapsed Time
Get 1588 time equal to LXI
Get Lxi time converted to local
Get string for Lxi time (local)
Get string forLxi time
Get UTC time
uses IDs from
LxiEventID
ALL
LAN0 - LAN7
LxiERROR
Receives any Lxi Event
Lxi Standard Event
LxiEvent Error flags
LxiEventFlags (enumeration)
None
Error
Retransmission
HWSlope
Acknowledgement
No flags are set
Error Flag
Retransmission
HW slope
Acknowledgement
© Copyright Agilent Technologies 2007

PTP (LXI Timing) Namespaces

Classes

MessageReceivedEventArgs	Event arguments sent to the event handler when an Ptp Sync, Announce or Followup message is received.
ptpAnnounceMessage	Provides access to PTP ANNOUNCE message elements.
ptpClock	PtpClock contains clock state information. Enables clock configuration.
ptpClockHistory	Stores the timestamp of each sync packet and the value of the CurrentDataSet of the clock when that sync packet was received.
PtpClockQuality	Elements which describe the clock quality.
ptpCurrentData	The CURRENT DATA SET of a PTP clock. This data set characterizes dynamic properties of the clock.
ptpDefaultData	The DEFAULT DATA SET of a PTP clock. This data set defines inherent or assumed properties of the clock. These values are used when the clock becomes the grandmaster clock in a subdomain.
ptpFollowUpMessage	Provides access to PTP FOLLOW_UP message elements.
PtpManagementErrorStatus	Provides access to PTP management error status message elements.
PtpManagementMessage	Provides access to PTP managment message elements.
ptpManager	The manager for sending and receiving LXI timing information.

ptpMessage	Provides access to PTP common message header elements.
ptpParentData	The PARENT DATA SET of a PTP clock. If the clock has a port in the PTP_SLAVE state its master clock becomes its parent and the data received in SYNC and ANNOUNCE messages from that parent/master is recorded here.
PtpPortAddress	Represents the protocol or physical address of a PTP port.
ptpPortData	The PORT DATA SET of a PTP clock. This data set contains information specific to the PTP clock port. Ordinary clocks have one port. Boundary clocks have multiple ports.
PtpPortIdentity	The PortIdentity type identifies a PTP port.
ptpSyncMessage	Provides access to PTP SYNC message elements.
ptpTime	Contains the Ptp representation of time.
ptpTimePropertiesData	The TIME PROPERTIES DATA SET of a PTP clock. This data set contains information needed to interpret the time provided in PTP messages.

Interfaces

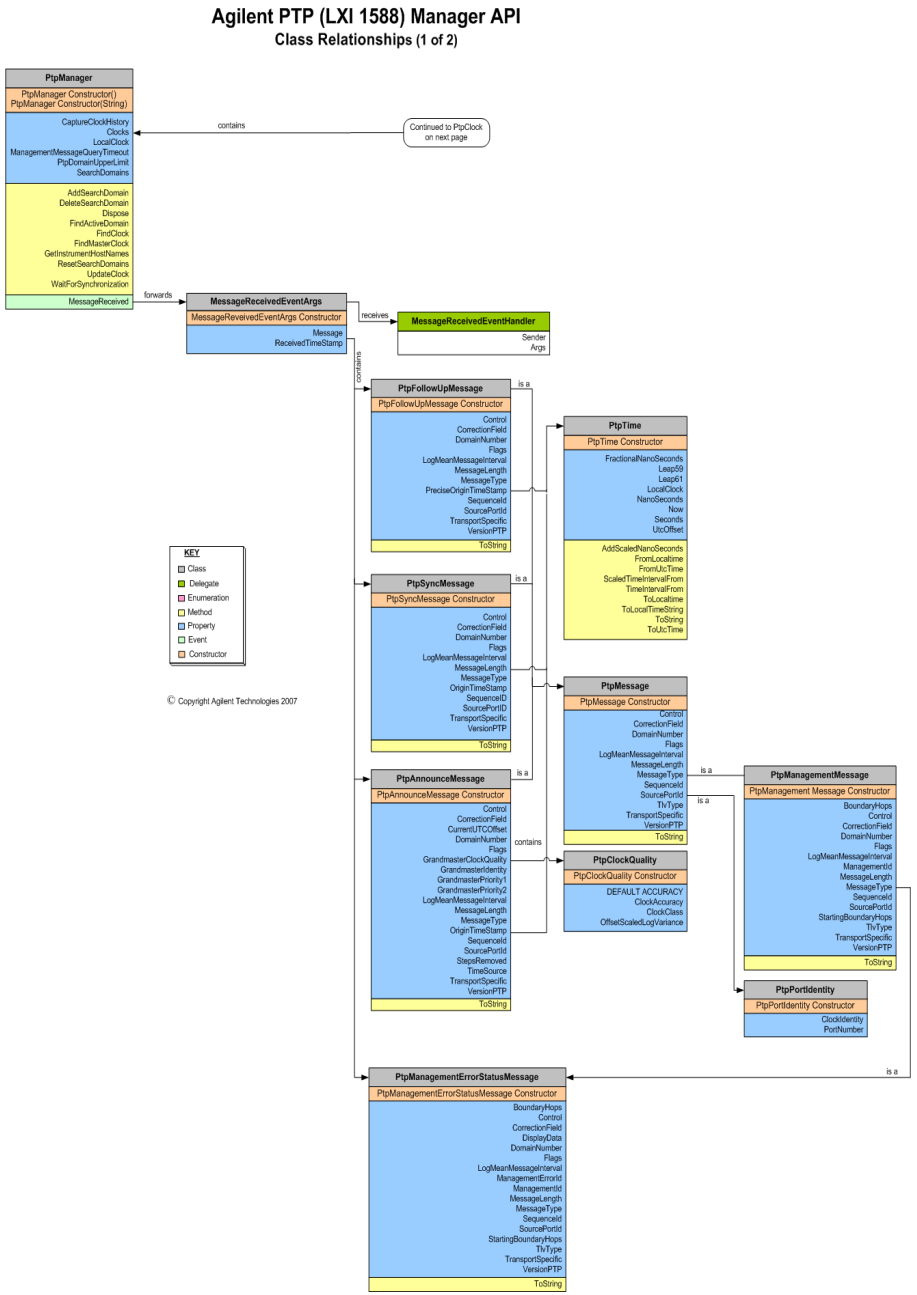
IPtpLocalClock	The interface for a local PTP clock, as used by PtpManager.

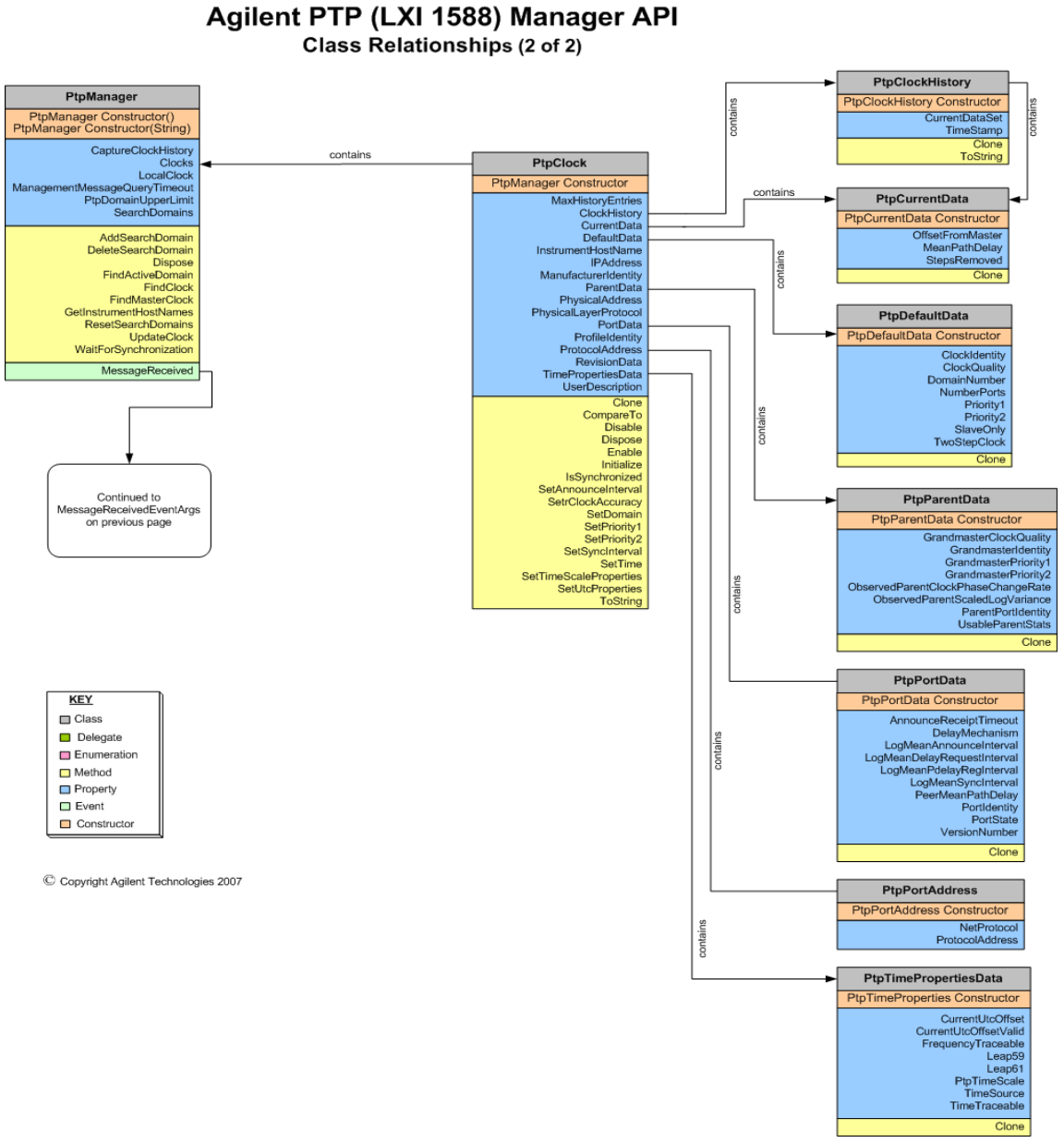
Delegates

MessageReceivedEventHandler	The event handler delegate definition for Ptp Sync packet received events.

Enumerations

ClockSyncStatus	Enumeration representing the synchronization status of a clock.
ControlField	The range of values found in the PtpMessageHeader.Control field. The use of this field by the receiver is deprecated. NOTE—This field is provided for compatibility with hardware designed to conform to version 1 of this standard.
MessageFlags	1588 Message Header flags
PtpClockAccuracy	Clock Accuracy values.
PtpClockType	Ptp Clock Types.
PtpDelayMechanism	Delay mechanism types for PTP clocks.
PtpManagementErrorId	Management Error Status Id enumeration, Management Error Status Message
PtpManagementId	The Management TLV types. Logically, subtypes of TLV
PtpMessageType	The range of values found in the PtpMessageHeader.MessageType field.
PtpNetProtocol	Underlying network protocol in use by the clock.
PtpPortState	States of the PTP finite state machine.
PtpTimeSource	Possible values for the grandmaster clocks's source of time.
PtpTlvType	TLV (Type, Length, Value) Types.


Agilent PTP (LXI 1588) Manager API
Class Relationships (1 of 2)
PtpManager
PtpManager Constructor()
PtpManager Constructor(String)
CaptureClockHistory
Clocks
LocalClock
ManagementMessageQueryTimeout
PtpDomainUpperLimit
SearchDomains
AddSearchDomain
DeleteSearchDomain
Dispose
FindActiveDomain
FindClock
FindMasterClock
GetInstrumentHostNames
ResetSearchDomains
UpdateClock
WaitForSynchronization
MessageReceived
contains
Continued to PtpClock on next page
forwards
MessageReceivedEventArgs
MessageReveivedEventArgs Constructor
Message
ReceivedTimeStamp
receives
MessageReceivedEventHandler
Sender
Args
contains
PtpFollowUpMessage
PtpFollowUpMessage Constructor
Control
CorrectionField
DomainNumber
Flags
LogMeanMessageInterval
MessageLength
MessageType
PreciseOriginTimeStamp
SequenceId
SourcePortId
TransportSpecific
VersionPTP
ToString
is a
PtpTime
PtpTime Constructor
FractionalNanoSeconds
Leap59
Leap61
LocalClock
NanoSeconds
Now
Seconds
UtcOffset
AddScaledNanoSeconds
FromLocaltime
FromUtcTime
ScaledTimeIntervalFrom
TimeIntervalFrom
ToLocaltime
ToLocalTimeString
ToString
ToUtcTime
KEY
Class
Delegate
Enumeration
Method
Property
Event
Constructor
PtpSyncMessage
PtpSyncMessage Constructor
Control
CorrectionField
DomainNumber
Flags
LogMeanMessageInterval
MessageLength
MessageType
OriginTimeStamp
SequenceID
SourcePortID
TransportSpecific
VersionPTP
ToString
is a
PtpMessage
PtpMessage Constructor
Control
CorrectionField
DomainNumber
Flags
LogMeanMessageInterval
MessageLength
MessageType
SequenceId
SourcePortId
TlvType
TransportSpecific
VersionPTP
ToString
is a
PtpManagementMessage
PtpManagement Message Constructor
BoundaryHops
Control
CorrectionField
DomainNumber
Flags
LogMeanMessageInterval
ManagementId
MessageLength
MessageType
SequenceId
SourcePortId
StartingBoundaryHops
TlvType
TransportSpecific
VersionPTP
ToString
PtpAnnounceMessage
PtpAnnounceMessage Constructor
Control
CorrectionField
CurrentUTCOffset
DomainNumber
Flags
GrandmasterClockQuality
GrandmasterIdentity
GrandmasterPriority1
GrandmasterPriority2
LogMeanMessageInterval
MessageLength
MessageType
OriginTimeStamp
SequenceId
SourcePortId
StepsRemoved
TimeSource
TransportSpecific
VersionPTP
ToString
is a
contains
PtpClockQuality
PtpClockQuality Constructor
DEFAULT ACCURACY
ClockAccuracy
ClockClass
OffsetScaledLogVariance
PtpPortIdentity
PtpPortIdentity Constructor
ClockIdentity
PortNumber
PtpManagementErrorStatusMessage
PtpManagementErrorStatusMessage Constructor
BoundaryHops
Control
CorrectionField
DisplayData
DomainNumber
Flags
LogMeanMessageInterval
ManagementErrorId
ManagementId
MessageLength
MessageType
SequenceId
SourcePortId
StartingBoundaryHops
TlvType
TransportSpecific
VersionPTP
ToString
is a
© Copyright Agilent Technologies 2007


Agilent PTP (LXI 1588) Manager API
Class Relationships (2 of 2)
PtpManager
PtpManager Constructor()
PtpManager Constructor(String)
CaptureClockHistory
Clocks
LocalClock
ManagementMessageQueryTimeout
PtpDomainUpperLimit
SearchDomains
AddSearchDomain
DeleteSearchDomain
Dispose
FindActiveDomain
FindClock
FindMasterClock
GetInstrumentHostNames
ResetSearchDomains
UpdateClock
WaitForSynchronization
MessageReceived
contains
PtpClock
PtpManager Constructor
MaxHistoryEntries
ClockHistory
CurrentData
DefaultData
InstrumentHostName
IPAddress
ManufacturerIdentity
ParentData
PhysicalAddress
PhysicalLayerProtocol
PortData
ProfileIdentity
ProtocolAddress
RevisionData
TimePropertiesData
UserDescription
Clone
CompareTo
Disable
Dispose
Enable
Initialize
IsSynchronized
SetAnnounceInterval
SetrClockAccuracy
SetDomain
SetPriority1
SetPriority2
SetSyncInterval
SetTime
SetTimeScaleProperties
SetUtcProperties
ToString
Continued to
MessageReceivedEventArgs
on previous page
PtpClockHistory
PtpClockHistory Constructor
CurrentDataSet
TimeStamp
Clone
ToString
PtpCurrentData
PtpCurrentData Constructor
OffsetFromMaster
MeanPathDelay
StepsRemoved
Clone
PtpDefaultData
PtpDefaultData Constructor
ClockIdentity
ClockQuality
DomainNumber
NumberPorts
Priority1
Priority2
SlaveOnly
TwoStepClock
Clone
PtpParentData
PtpParentData Constructor
GrandmasterClockQuality
GrandmasterIdentity
GrandmasterPriority1
GrandmasterPriority2
ObservedParentClockPhaseChangeRate
ObservedParentScaledLogVariance
ParentPortIdentity
UsableParentStats
Clone
PtpPortData
PtpPortData Constructor
AnnounceReceiptTimeout
DelayMechanism
LogMeanAnnounceInterval
LogMeanDelayRequestInterval
LogMeanPdelayReqInterval
LogMeanSyncInterval
PeerMeanPathDelay
PortIdentity
PortState
VersionNumber
Clone
PtpPortAddress
PtpPortAddress Constructor
NetProtocol
ProtocolAddress
PtpTimePropertiesData
PtpTimeProperties Constructor
CurrentUtcOffset
CurrentUtcOffsetValid
FrequencyTraceable
Leap59
Leap61
PtpTimeScale
TimeSource
TimeTraceable
Clone
KEY
Class
Delegate
Enumeration
Method
Property
Event
Constructor
© Copyright Agilent Technologies 2007

Glossary

Accuracy

The mean of the time or frequency error between a clock and the reference clock (such as the Master). Stability is the measure of how the mean varies over time, temperature, etc. Precision is a measure of the deviation of the error from the mean.

API

Application programming interface; a well-defined set of software routines through which an application program can access the functions and services provided by an underlying operating system or a reusable software library

Best Master Clock Algorithm

This is an IEEE 1588 specific term. PTP clocks attached to a network subnet organize themselves into a master/slave relationship using a 'best master clock' algorithm that determines the clock best suited to become the master clock. No central controller is required for this to happen. However, clocks have priority fields that a central controller can set to influence the outcome of the best master clock algorithm.

Boundary Clock or Boundary Switch

This is an IEEE 1588 specific term. A boundary clock is a clock with more than a single PTP port, with each PTP port providing access to a separate PTP communication path. Boundary clocks eliminate fluctuations produced by routers and similar network elements.

Civil Time

Another name for mean solar time reckoned from midnight. Generally, it refers to specific standard time scales designated by civilian authorities or a reference to local or standard time indicated.

Clock

A node that is capable of providing a measurement of the passage of time since a defined epoch. In the IEEE 1588 standard, the term clock is interpreted to mean either an ordinary clock or a PTP port of a boundary clock unless otherwise stated or obvious from the context.

Clock Timestamp Point

IEEE 1588 requires the generation of a timestamp on transmission or receipt of all 1588 **Sync** and **Delay_Req** messages. The point in the outbound and inbound protocol stacks where this timestamp is generated is called the clock timestamp point.

Data Set

This is an IEEE 1588 specific term. Data sets define inherent or assumed properties of a local PTP clock. There are two kinds of Data Sets, Fixed and Modifiable. Fixed Data Sets are inherent properties of the clock. Modifiable Data Sets may change and are initialized at power-up, in response to an initalization event or management message.

Device Class

The LXI standard defines three types of instruments that you can easily mix and match within a test system; these are defined as Class A, Class B, and Class C. Refer to "LXI Device Classes" on page 10 for detailed information.

Direct Communication

The communication of PTP information between two PTP clocks with no intervening boundary clock is termed a direct communication.

Domain

A group of devices managed by a single directory and using shared resources. The LXI domain runs from 0 - 255, the PTP domain runs from 0-127. They both group LXI instruments or PTP clocks. Users should make sure that LXI instruments and their clocks are grouped similarly with each domain type.

End-to-End Transparent Switch

A PTP Transparent Switch that corrects PTP event messages for the time that the message resided in the Transparent Switch. This is an IEEE 1588-2007 clock device type not implemented in the IEEE 1588-2002 spec.

Epoch

The reference time defining the origin or start of a time scale. The IEEE 1588 epoch is 01 January, 1970 @ 00:00:00 TAI.

Event

Also known as LXI Event. A signal (such as a trigger signal) or announcement of a state change. An abstraction of the mechanism by which signals or conditions are generated and represented.

Event Edge

The “edge” of an instrument or LXI event (such as a hardware trigger) that is encoded into a transmitted LXI packet as a binary flag representing the rising edge (1) or falling edge (0) of the state transition. This is also known as Event Polarity, Event Sense, or Event Detection.

Event ID

Also known as LXI Event ID. A 16-byte Event Identifier. Note that some event names (such as LAN0 through LAN7) are predefined by the LXI Consortium. Other Event ID names are user definable.

External Synchronization

This is an IEEE 1588 specific term. It is often desirable to synchronize a single clock to an external source of time, for example to a GPS system to establish a UTC time base. This synchronization is accomplished by means other than those specified by 1588 and is referred to as external synchronization

Global Positioning System (GPS)

GPS is the atomic time scale implemented by the atomic clocks in the GPS system (both ground control stations and satellites).

Grandmaster Clock

This is an IEEE 1588 specific term. Within a collection of IEEE 1588 clocks on a subnet, one clock, the Grandmaster Clock, serves as the primary source of time to which all others are ultimately synchronized. **Master clock**: A system of 1588 clocks may be segmented into regions separated by boundary clocks. Within each region there will be a single clock, the Master Clock, serving as the primary source of time. These Master Clocks will in turn synchronize to other Master Clocks and ultimately to the Grandmaster Clock. **Ordinary clock**: An ordinary clock is a 1588 clock with a single PTP port.

IEEE 1588 Precision Time Protocol

IEEE standard defining a protocol for enabling precise synchronization of clocks in a measurement and control system implemented with technologies such as network communication, local computing and distributed objects.

Julian Day

The integer number of days that have elapsed since the initial epoch defined as noon Universal Time on Monday, January 1, 4713 BC. That noon-to-noon day is counted as Julian day 0.

Latency

Latency is the time interval between when an event occurs or is transmitted to when it is detected or received.

Leap Second

UTC is occasionally adjusted by one second increments to ensure that the difference between a uniform time scale defined by atomic clocks does not differ from the Earth's rotational time by more than 0.9 seconds.

LXI

LAN eXtensions for Instrumentation; an instrumentation platform based on widely used standards such as Ethernet, TCP/IP and IVI (C or COM) drivers; small, faceless modules designed for use in PC-based automated test systems.

LXI Class

The LXI standard defines three classes of instruments:

LXI Class A The devices in this category satisfy the Class C and B requirements, and add the LXI trigger bus.

LXI Class B The devices in this category are designed to enable distributed measurement systems. They satisfy Class C requirements and include PTP time synchronization, time-based triggering, peer-peer LAN messages, and LAN event-based triggering.

LXI Class C The devices in this category are standalone or bench-type instruments that replace GPIB and support basic LAN connectivity. They support a web interface for instrument set-up and data access, viewable from a standard web browser. They also provide an IVI driver API.

Management Message

Management messages may be directed to all clocks or a specific clock within a subdomain. There are several defined Management MEssages such as Enable/Disable clock port, set sync interval, obtain clock identity, etc.

Management Node

A PTP device that: has one or more physical connections to the network, serves as a human or programmatic interface to PTP management messages, and may be combined with any clock type. Note: PtpManager allows a test PC to be a management node.

Master Clock

In the context of a single PTP communication path, a clock that when viewed from the path appears to be the source of time to which all other clocks synchronize. Also known as Parent Clock.

Modified Julian Day (MJD)

Modified Julian Date (MJD), abbreviated version of Julian Day (JD) dating method used for centuries by astronomers, geophysicists, chronologers, and others who needed to have an unambiguous dating system based on continuing day counts.

Multicast Communication

IEEE 1588 requires that PTP messages be communicated via a multicast. In this style of communication any node may post a message and all nodes in the same segment of a subdomain will receive this message. Boundary clocks define the segments within a subdomain.

Network Time Protocol (NTP)

Network Time Protocol is a hierarchical protocol for synchronizing the clocks of computer systems over packet-switched, variable-latency data networks. NTP uses UDP port 123 as its transport layer. The assigned multicast IP address is 224.0.1.1 - but NTP may use other IP addresses as well.

One-step Clock

A PTP ordinary or boundary clock accurately sets the originTimestamp and correction fields of the event messages that it transmits. A PTP transparent clock accurately updates the correction field of the event messages that it forwards.

Ordinary Clock

An ordinary clock is an IEEE 1588 clock with a single PTP port.

Packet

Also known as LXI Packet. The specific format of network packet that carries an LXI message over the network.

Parent Clock

Synonymous with Master Clock.

Peer-to-Peer Transparent Switch

A PTP Transparent Switch that corrects PTP event messages for the time that the message resided in the Transparent Switch and in addition corrects for the propagation time on the inbound link. This is an IEEE 1588-2007 clock device type not implemented in the IEEE 1588-2002 spec.

Port

A port is an abstraction of point to point communication for transmitting and receiving messages over a computer network. It is an index identifying a specific PTP port on a PTP clock. Port states determining the Master-Slave hierarchy are:

- Master -- The port is the source of time on the network served by the port.
- Slave -- The port synchronizes to the device in the domain that is the Master.
- Passive -- The port is not the Master nor a Slave and it does not synchronize to the Master.

Precision Time Protocol (PTP)

PTP is the acronym for Precision Time Protocol, the name used in the IEEE 1588 standard for the protocol that enables precise synchronization of nodes on a computer network. It is more precise than NTP but less costly than GPS.

Preferred Master Clock

A clock may be administratively designated as a preferred Master Clock to specify a clock that will remain master in the presence of disconnection or connection of other clocks.

Preferred Master Clock Set

A set of clocks that will be favored over those not so designated in the selection of the Grandmaster Clock within a subdomain.

PTP Clock

An ordinary clock that participates in the PTP protocol.

PTP Domain

A set or collection of PTP subdomains. A Domain may have several subdomains; clocks within a subdomain synchronize with other clocks in the same subdomain but not necessarily clocks in other subdomains even though they are all within the same larger Domain.

PTP Message

There are ten designated messages types defined by IEEE 1588: **Sync**, **Delay_Req**, **Follow-up**, **Delay_Resp**, **Pdelay_Req**, **Pdelay_Resp**, **Pdelay_Resp_Follow_Up**, **Announce**, **Signaling**, and **Management**.

PTP Port

A PTP port is the logical access point for IEEE 1588 communications to the clock containing the port.

PTP Subdomain

A logical grouping of PTP clocks that synchronize to each other using the PTP protocol but that are not necessarily synchronized to PTP clocks in another PTP subdomain.

PTP Timescale

The IEEE 1588 specification uses the term "time scale" to indicate whether the time kept by PTP clocks is synchronized with the PTP epoch.

PTP Timestamp

An event timestamp that was generated by the PTP clock and is in the TAI time scale. It typically is generated from a hardware timestamp that is converted to the TAI time.

Stratum

In NTP, stratum levels define the distance from the reference clock. Stratum is not used in IEEE 1588 2007.

Stratum-1 Time Server

A time server directly linked (not over a network path) to a reliable source of UTC time such as GPS, WWV, or CDMA transmissions. A stratum-1 time server acts as a primary network time standard.

Synchronized Clocks

Two clocks are synchronized to a specified uncertainty if they have the same epoch, and measurements of any time interval by both clocks differ by no more than the specified uncertainty. The timestamps generated by two synchronized clocks for the same event will differ by no more than the specified uncertainty.

Sync Interval

This is an IEEE 1588 specific term. The interval, in seconds, between successive Sync messages issued by the Master Clock. The sync interval is a compromise between the inherent stability of the clocks, the responsiveness of the clocks in a subdomain to change, and the communication load imposed by PTP.

Sync Threshold

This is an IEEE 1588 specific term. The maximum Slave Clock offset allowed in a test system where the Slave is considered to be “in sync” with the Master Clock. For example, in one test system a threshold of 25 nS (the Slave may be ± 25 nS from the Master) may be considered in sync but a different system may only require a threshold of ±1 µS.

Synthetic Instruments

A collection of hardware and software modules combined to emulate standard instruments. Hardware and software modules can be reconfigured to provide needed functionality, thus reducing redundant components. Synthetic instruments are used primarily in aerospace/defense applications.

Syntonized Clocks

Two clocks are syntonized if they share the same definition of “second”, that is, time is measured at the same rate. They may or may not share the same epoch.

System Time

Absolute time value in the context of a distributed time system where all devices have a local clock that is synchronized with a common Master Clock. System Time is a 64-bit integer value in units of nanoseconds or microseconds with a value of 0 corresponding to an epoch of January 1, 1970.

TAI

International Atomic Time (TAI, from the French Temps Atomique International) is a high-precision atomic time standard.

Time Scale

A linear measure of time from an epoch.

Time Stamp

The time at which an event occurs, added to a message packet. It is defined by the instant a message timestamp point passes the clock timestamp point in a protocol stack.

Time Server

A Time Server provides the correct, current time to clients when requested. This role can be assumed by time servers at various strata.

Transparent Switch or Transparent Clock

A switch that compensates for its own queueing delays. Unlike a Boundary Clock, a Transparent Switch has neither Master nor Slave Clock capabilities.

Two-step Clock

A PTP ordinary or boundary clock does not accurately set the originTimestamp and correction fields of the event messages that it transmits. A PTP transparent clock does not update the correction field of the event messages that it forwards. The accurate information is transmitted in a subsequent general message.

UDP

Abbreviation for User Datagram Protocol. This is a core Internet protocol that provides for the transmission of short messages without guaranteed delivery. It is especially useful as a multicast mechanism for sending a common message simultaneously to a list of subscribers.

UTC

Coordinated Universal Time (UTC) is a high-precision atomic time standard.

UUID

UUID is an acronym for Universally Unique Identifier, an identifier that has a unique value within some defined universe. For purposes of the IEEE 1588 standard, the universe consists of all possible PTP devices having a UUID unless otherwise stated.

Index

T

U

V

W