Eaton

PowerChain Toolkit Time Synchronization and Library Specification

Copyright © 2005-2007 Eaton Corporation. All Rights Reserved.
NOTICE TO PERSONS RECEIVING THIS DRAWING AND/OR TECHNICAL INFORMATION

THIS DRAWING AND/OR TECHNICAL INFORMATION IS THE PROPERTY OF EATON CORPORATION

APPROVED BY/DATE

<table>
<thead>
<tr>
<th>DATE</th>
<th>SIGNATURE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initiated By:</td>
<td></td>
</tr>
<tr>
<td>Prepared By:</td>
<td></td>
</tr>
<tr>
<td>Approved By:</td>
<td></td>
</tr>
<tr>
<td>Approved By:</td>
<td></td>
</tr>
<tr>
<td>Approved By:</td>
<td></td>
</tr>
<tr>
<td>Approved By:</td>
<td></td>
</tr>
</tbody>
</table>
## 1 Revision History

<table>
<thead>
<tr>
<th>Rev.</th>
<th>Date</th>
<th>By</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>09/21/05</td>
<td>Chris Egstad</td>
<td>Draft</td>
</tr>
<tr>
<td>1.0</td>
<td>06/07/06</td>
<td>Chris Egstad</td>
<td>Initial Release</td>
</tr>
<tr>
<td>1.1</td>
<td>06/09/06</td>
<td>Chris Egstad</td>
<td>Updated trace definitions</td>
</tr>
<tr>
<td>1.2</td>
<td>09/06/06</td>
<td>Chris Egstad</td>
<td>Documented configuration modifications to NTPD and API additions to the Time Library.</td>
</tr>
<tr>
<td>1.3</td>
<td>04/09/07</td>
<td>Chris Egstad</td>
<td>Added notes for the toolkit custom NTP modifications and embedded application usage.</td>
</tr>
<tr>
<td>1.4</td>
<td>04/15/07</td>
<td>Chris Egstad</td>
<td>Updated notes for the toolkit custom NTP modifications</td>
</tr>
<tr>
<td>1.41</td>
<td>04/20/07</td>
<td>Chris Egstad</td>
<td>Slight change to example code in 6.6.and 6.7</td>
</tr>
</tbody>
</table>
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 REVISION HISTORY</td>
<td>2</td>
</tr>
<tr>
<td>1 INTRODUCTION</td>
<td>5</td>
</tr>
<tr>
<td>1.1 Basic Requirements for Time Library and NTP Time Synchronization</td>
<td>5</td>
</tr>
<tr>
<td>1.1.1 RFC-1305 Support</td>
<td>5</td>
</tr>
<tr>
<td>1.1.2 Client Side Implementation</td>
<td>5</td>
</tr>
<tr>
<td>1.1.3 Time Library</td>
<td>5</td>
</tr>
<tr>
<td>1.1.4 Timestamps and UTC</td>
<td>5</td>
</tr>
<tr>
<td>2 TIME SYNCHRONIZATION IMPLEMENTATION</td>
<td>6</td>
</tr>
<tr>
<td>2.1 NTP SOFTWARE DISTRIBUTION</td>
<td>6</td>
</tr>
<tr>
<td>2.1.1 Network Time Protocol Version 4 Distribution</td>
<td>6</td>
</tr>
<tr>
<td>2.1.2 NTP Version</td>
<td>6</td>
</tr>
<tr>
<td>2.1.3 NTPD (NTP Daemon)</td>
<td>6</td>
</tr>
<tr>
<td>2.1.4 OpenSSL Support</td>
<td>6</td>
</tr>
<tr>
<td>2.1.5 NTP Utility Programs</td>
<td>6</td>
</tr>
<tr>
<td>3 PCD TOOLKIT MODIFICATIONS FOR NTPD</td>
<td>7</td>
</tr>
<tr>
<td>3.1 Saving Command-Line Options to the NTP Configuration File</td>
<td>7</td>
</tr>
<tr>
<td>3.1.1 Save Configuration Rules for Custom Options</td>
<td>7</td>
</tr>
<tr>
<td>3.1.2 PCD Toolkit Custom Command-Line Options for NTPD</td>
<td>8</td>
</tr>
<tr>
<td>3.2 NTPD Status Change Messages Written to the CDS</td>
<td>9</td>
</tr>
<tr>
<td>3.3 Large NTP Clock Steps Initiate a CDS Set Operation</td>
<td>9</td>
</tr>
<tr>
<td>3.4 NTPD Panic Threshold Test is Always Disabled</td>
<td>9</td>
</tr>
<tr>
<td>4 TIME LIBRARY API</td>
<td>10</td>
</tr>
<tr>
<td>4.1 Timestamp Functions</td>
<td>10</td>
</tr>
<tr>
<td>4.1.1 Eaton Timestamp Format</td>
<td>10</td>
</tr>
<tr>
<td>4.1.2 etn.UTCtime()</td>
<td>10</td>
</tr>
<tr>
<td>4.1.3 etn.Localtime()</td>
<td>10</td>
</tr>
<tr>
<td>4.1.4 etn.UNIXtoETNtime()</td>
<td>10</td>
</tr>
<tr>
<td>4.1.5 etn.ETNtoUNIXtime()</td>
<td>10</td>
</tr>
<tr>
<td>4.1.6 etn.UTCtime64()</td>
<td>10</td>
</tr>
<tr>
<td>4.1.7 etn.Localtime64()</td>
<td>11</td>
</tr>
<tr>
<td>4.1.8 etn.UNIXtoETNtime64()</td>
<td>11</td>
</tr>
<tr>
<td>4.1.9 etn.ETNtoUNIXtime64()</td>
<td>11</td>
</tr>
<tr>
<td>4.1.10 etn.ETNtoUNIXtime64()</td>
<td>11</td>
</tr>
<tr>
<td>4.2 Time Zone Functions</td>
<td>12</td>
</tr>
<tr>
<td>4.2.1 etn.GetLocalTimeZone()</td>
<td>12</td>
</tr>
<tr>
<td>4.2.2 etn.GetLocalTimeZoneName()</td>
<td>12</td>
</tr>
<tr>
<td>4.2.3 etn.IsDaylightSavings()</td>
<td>12</td>
</tr>
<tr>
<td>4.3 Trace Functions</td>
<td>13</td>
</tr>
<tr>
<td>4.3.1 etn.NTPtrace()</td>
<td>13</td>
</tr>
<tr>
<td>4.3.2 etn.IPtrace()</td>
<td>13</td>
</tr>
<tr>
<td>5 TIME ZONE AND DAYLIGHT SAVINGS</td>
<td>15</td>
</tr>
<tr>
<td>6 NTPD USAGE NOTES FOR THE PXG ARM 9 LINUX SYSTEM</td>
<td>16</td>
</tr>
<tr>
<td>6.1 Kernel Clock Tick</td>
<td>16</td>
</tr>
</tbody>
</table>
1 Introduction

As complicated events occur throughout the PowerChain the need to coordinate the time accuracy across various devices and software components becomes more and more critical. This specification outlines the TAR (Timing Accuracy Requirements) for product development.

It is unrealistic to assume that each device will accommodate either its own GPS or Rubidium timing source, will have its own proprietary cable or contain the other costs associated with high accuracy devices. This drives the need for a centralized strategy across both hardware and software networked devices.

Although there is a question of accuracy regarding NTP, the latest version (versions 3 and 4) significantly increase accuracy across a network and is truly the most practical (tried and tested) method for distributing an accurate timing source to remote devices. NTP will compensate for network latency by using an algorithm to calculate when an NTP packet was received and sent. Direct Ethernet connections to a Stratum I Master Clock yield normal accuracy times of +/-50 microseconds to UTC. The reader is referenced to: http://www.faqs.org/rfcs/rfc1305.html

As part of the PowerChain Device (PCD) Toolkit, this document describes the requirements and specifications for Eaton PowerChain Devices implementing NTP Time Synchronization.

In addition, this document also describes the requirements and API specifications for the PowerChain Toolkit Time Library.

1.1 Basic Requirements for Time Library and NTP Time Synchronization

1.1.1 RFC-1305 Support
Software implementations will be on the client-side of the client-server NTP model. It will be assumed that a Master Clock and NTP server exists at an IP address known by the client.

1.1.2 Client Side Implementation
In order to support the widest variety of established timing systems that may already be installed at a user’s site, the client-side Time Synchronization implementation will use NTP, RFC 1305 Version 3 or above.

1.1.3 Time Library
The Time Library will be included in the PowerChain Toolkit, so that device re-use is encouraged wherever the Toolkit is applied. This toolkit will set the local device’s clock.

1.1.4 Timestamps and UTC
Timestamps will be relative to UTC, thus, avoiding any daylight saving or time zone issues.
2 Time Synchronization Implementation

The Time Synchronization implementation will be derived from a standard (or near-standard) implementation of the v4.2.4p0 NTPD daemon from the “Network Time Protocol Version 4 Distribution”. This will satisfy the Toolkit requirements stated in sections 1.1.1 and 1.1.2.

To keep binary size to a minimum on embedded targets, the toolkit NTPD daemon will be compiled with only the required clock drivers.

2.1 NTP Software Distribution

2.1.1 Network Time Protocol Version 4 Distribution

The NTP software package distribution can be found at http://www.ntp.org “Home of the Network Time Protocol project”. This source code will be used for building the NTPD daemon, and will serve as a base for implementing several Time Library functions. This code is available under a BSD license and can be modified without revealing source code, although the NTP copyright notice (see section 6.0) must accompany any binary distribution or installation. There is no RFC yet for the v4 NTP specification.

2.1.2 NTP Version

The NTP code base is currently at release version 4.2.4p0, and supports a wide range of UNIX systems as well as Windows NT, 200, and XP (NT services). Windows 2003 support is not explicitly mentioned in the documentation, but several NTP software download sites refer to Windows 2003 as being supported (subject to confirmation). This code supports version 4 of the NTP specification, but is backwards compatible with NTP v3. An important note about NTP version 3 distributions is they do not support Windows as well as the version 4 distribution.

2.1.3 NTPD (NTP Daemon)

The NTPD daemon will be installed on any system requiring NTP synchronization using default poll and update values (TBD). The daemon will run in the Client (3) mode of operation, and will (by default) use the Type 1 Undisciplined Local Clock (LOCAL) to create a Reference Clock. Section 4 defines the PCD Toolkit modifications and enhancements to the standard NTP daemon.

2.1.4 OpenSSL Support

The NTPD daemon includes OpenSSL support.

2.1.5 NTP Utility Programs

The NTP utilities listed below are built as part of the standard Toolkit NTP build, but are not required to be deployed on a target platform.

- ntpq - standard NTP query program.
- ntpdc - special NTP query program.
- ntpdate - set the date and time via NTP.
- tickadj - set time-related kernel variables.
3 PCD Toolkit Modifications for NTPD

3.1 Saving Command-Line Options to the NTP Configuration File

The standard NTPD daemon does not save command-line configuration options to the NTP configuration file (ntp.conf). The PCD Toolkit NTPD daemon has been customized to save a select list of custom command-line options to the NTP configuration file.

3.1.1 Save Configuration Rules for Custom Options

1) The default NTP configuration file location is:

\[ /etc/ntp.conf \]

2) Invoking the standard NTPD ‘–c’ option with alternate configuration file name and path will save any of the defined "Save Command-Line Options" to the different configuration file location.

3) Invoking any of the defined "Save Command-Line Options" will update the configuration file.

4) Option parameters MUST be bracketed by double quotes (").

\[
\text{OK: } \texttt{./ntpd -H "restrict 128.4.0.0 mask 255.255.0.0 \#Allow DCnet clients"} \\
\text{NOT OK: } \texttt{./ntpd -H restrict 128.4.0.0 mask 255.255.0.0 \#Allow DCnet clients}
\]

Note that the '#Allow DCnet clients' comment text will also be added to the configuration file.

5) An option parameter may be invoked multiple times.

For example, to add a pool of servers to the 'ntp.conf' file, the following command-line format could be used:

\[
\texttt{./ntpd -S "0.us.pool.ntp.org" -S "1.us.pool.ntp.org" -S "2.us.pool.ntp.org"}
\]

6) Invoking an option will delete all instances of that option from the current configuration file. For example, if the current 'ntp.conf' file has the following "restrict" options:

\[
\text{restrict default noserve} \\
\text{restrict 127.0.0.1} \\
\text{restrict 128.4.0.0 mask 255.255.0.0}
\]

Then after invoking the following "restrict" command-line option,

\[
\texttt{./ntpd -H "128.175.0.0"}
\]

The new 'ntp.conf' file would contain only one "restrict" option:

\[
\text{restrict 128.175.0.0}
\]
### 3.1.2 PCD Toolkit Custom Command-Line Options for NTPD

See the following NTP documentation files for more specific information on options.

- accept.html
- confopt.html
- clockopt.html
- manyopt.html
- miscopt.html
- monopt.html
- notes.html

#### Figure 3-1 PCD NTPD Command-Line Options

<table>
<thead>
<tr>
<th>Option Name</th>
<th>Command-line Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. autokey</td>
<td>-0 &quot;[logsec]&quot;</td>
</tr>
<tr>
<td>2. broadcast</td>
<td>-1 &quot;address [options ...]&quot;</td>
</tr>
<tr>
<td>3. broadcastclient</td>
<td>-2 &quot;[novolley]&quot;</td>
</tr>
<tr>
<td>4. broadcastdelay</td>
<td>-r &quot;seconds&quot;</td>
</tr>
<tr>
<td>5. calldelay</td>
<td>-0 &quot;delay&quot;</td>
</tr>
<tr>
<td>6. crypto</td>
<td>-3 &quot;[cert file] [leap file] [randfile file] [host file] [sign file] [ident scheme] [ifppar file] [gpar file] [mvpar file] [pw password]&quot;</td>
</tr>
<tr>
<td>7. driftfile</td>
<td>-f &quot;driftfile [ minutes [ tolerance ] ]&quot;</td>
</tr>
<tr>
<td>8. filegen</td>
<td>-5 &quot;name [file filename] [type typename] [link</td>
</tr>
<tr>
<td>9. fudge</td>
<td>-7 &quot;127.127.t.u [time1 sec] [time2 sec] [stratum int] [refid string] [mode int] [flag1 0</td>
</tr>
<tr>
<td>10. logfile</td>
<td>-l &quot;logfile&quot;</td>
</tr>
<tr>
<td>11. keys</td>
<td>-k &quot;keyfile&quot;</td>
</tr>
<tr>
<td>12. manycastclient</td>
<td>-8 &quot;address [ ...]&quot;</td>
</tr>
<tr>
<td>13. manycastserver</td>
<td>-9 &quot;address [ ...]&quot;</td>
</tr>
<tr>
<td>14. multicastclient</td>
<td>-B &quot;address [ ...]&quot;</td>
</tr>
<tr>
<td>15. peer</td>
<td>-C &quot;address [options ...]&quot;</td>
</tr>
<tr>
<td>16. restrict</td>
<td>-H &quot;address [mask mask] [flag][...]&quot;</td>
</tr>
<tr>
<td>17. server</td>
<td>-S &quot;address [options ...]&quot;</td>
</tr>
<tr>
<td>18. statistics</td>
<td>-T &quot;name [ ...]&quot;</td>
</tr>
<tr>
<td>19. statsdir</td>
<td>-s &quot;directory_path&quot;</td>
</tr>
<tr>
<td>20. tinker</td>
<td>-Z &quot;[allan allan]</td>
</tr>
<tr>
<td>21. tos</td>
<td>-W &quot;[ ceiling ceiling</td>
</tr>
<tr>
<td>22. ttl</td>
<td>-Y &quot;hop ...&quot;</td>
</tr>
</tbody>
</table>

23. Exit ntpd after saving updates to ntp.conf file

- E "" (note that the dummy "" argument after the '-E' is required to prevent the NTP command-line parser from throwing an error)
3.2 NTPD Status Change Messages Written to the CDS

The Toolkit version of the NTPD daemon will write a message string to the Common Data Store (CDS) when the NTPD status changes. The following string values (in bold) will be written to the NTP status path location “/Software/Data Servers/NTP/Status/Value” as an indication of NTPD status.

1) "Started, not synchronized" – Set when NTPD first begins. Indicates NTPD is running, but is not synchronized with an NTP server, or controlling the system clock.
2) “Synchronizing with NTP server” – Set when NTPD has synchronized with the Local Clock, but is not yet synchronized with an NTP server.
3) “Synchronized to XX, NTP stratum YY” – Set when NTPD has synchronized with the NTP server XX, which is operating at NTP Stratum YY.
4) "Not running" – Set when NTPD exits. Indicates NTPD is not running.
5) "Stopped. Time correction exceeded sanity limit; set clock manually" – Error message indicating NTP cannot step the time far enough. Should not happen in practice, but is included for completeness.
6) "Kernel time sync error XX" – Error message indicating NTP cannot set the kernel clock. Should not happen in practice, but is included for completeness.

Toolkit applications using the CDS may register an event to this path when they want to be immediately informed of a change in NTP status. See section 6.5 for a C++ code example.

The CONFIG_EATON_CDSUTIL environment variable must be set to ‘y’ and exported prior to building the NTP project for this custom code to be included into the NTPD daemon. If testing with a white box system, make sure that there is an executable path set to the location of the ‘cdsutil’ program.

3.3 Large NTP Clock Steps Initiate a CDS Set Operation

When NTP steps the system clock by more than ±1 second, the Toolkit version of the NTP daemon will set the CDS variable “/Software/Data Servers/NTP/ClockStep/Value” to “true” (a Boolean value). Toolkit applications using the CDS may register an event to this path when they want to be immediately informed of a large change to the system clock time. See section 6.6 for a C++ code example.

The CONFIG_EATON_CDSUTIL environment variable must be set to ‘y’ and exported prior to building the NTP project for this custom code to be included into the NTPD daemon. If testing with a white box system, make sure that there is an executable path set to the location of the ‘cdsutil’ program.

3.4 NTPD Panic Threshold Test is Always Disabled

The ‘-g’ option is supposed to allow NTP to set the system clock ahead or behind a large amount without throwing a panic threshold error, and exiting the program. However, testing showed that this was not always the case, especially when using the Local Clock options “server 127.127.1.0” and “fudge 127.127.1.0 stratum 10” in the NTP.CONF file. In some cases, NTPD would generate the system error message “time correction of X.Y seconds exceeds sanity limit; set clock manually to the correct UTC time”, and the program would exit.

To prevent this from occurring in embedded applications, the NTPD panic threshold test has been disabled in the Toolkit version of the NTP daemon.
# 4 Time Library API

The following sections provide an overview of the definition and format of Time Library API functions.

## 4.1 Timestamp Functions

### 4.1.1 Eaton Timestamp Format

The Eaton Timestamp format and storage structure is equivalent to the Windows `FILETIME` structure, which is a 64-bit value representing the number of 100-nanosecond intervals since January 1, 1601 UTC. The definition of the Eaton equivalent structure `etn_time_t` is:

```c
typedef struct _etn_time_t
{
    uint32_t dwLowDateTime;   /* low 32 bits */
    uint32_t dwHighDateTime;  /* high 32 bits */
} etn_time_t;
```

### 4.1.2 `etn_UTCtime()`

Retrieves the current UTC Time in `etn_time_t` format. The formal definition of `etn_UTCtime()` is:

```c
/**
 * @param pUTCtime Pointer to struct etn_time_t.
 * @return  A non-zero return value indicates an error condition.
 */
int etn_UTCtime(etn_time_t *pUTCtime);
```

### 4.1.3 `etn_Localtime()`

Retrieves the current Local Time in `etn_time_t` format. The formal definition of `etn_Localtime()` is:

```c
/**
 * @param pLocaltime Pointer to struct etn_time_t.
 * @return A non-zero return value indicates an error condition.
 */
int etn_Localtime(etn_time_t *pLocaltime);
```

### 4.1.4 `etn_UNIXtoETNtime()`

Converts UNIX timestamp in a `timeval` format to Eaton `etn_time_t` format. The formal definition of `etn_UNIXtoETN()` is:

```c
/**
 * @param pUnix Pointer to timeval struct containing valid time data.
 * @param pETN Pointer to etn_time_t struct - will contain converted data.
 * @return A non-zero return value indicates an error condition.
 */
int etn_UNIXtoETN(timeval *pUnix, etn_time_t *pETN);
```

### 4.1.5 `etn_ETNtoUNIXtime()`

Converts Eaton timestamp in `etn_time_t` format to UNIX `timeval` format. The formal definition of `etn_ETNtoUNIX()` is:

```c
/**
 * @param pETN Pointer to etn_time_t struct containing valid time data.
 * @param pUnix Pointer to timeval struct - will contain converted data.
 * @return A non-zero return value indicates an error condition.
 */
int etn_ETNtoUNIX(etn_time_t *pETN, timeval *pUnix);
```

### 4.1.6 `etn_UTCtime64()`

Retrieves the current UTC Time as a uint64_t value. The formal definition of `etn_UTCtime64()` is:
Retrieves the current UTC Time as a uint64_t value. The formal definition of `etn_Localtime64()` is:

```c
/**
 * @return UTC Time as a uint64_t value.
 * A zero return value indicates an error condition.
 */
uint64_t etn_Localtime64(void);
```

**4.1.8 etn_UNIXtoETNtime64()**

Converts UNIX timeval format to a uint64_t value. The formal definition of `etn_UNIXtoETNtime64()` is:

```c
/**
 * @param pUnix Pointer to timeval struct containing valid time data.
 * @return ETN Time as a uint64_t value.
 * A zero return value indicates an error condition.
 */
uint64_t etn_UNIXtoETNtime64(timeval *pUnix);
```

**4.1.9 etn_ETNtoUNIXtime64()**

Converts Eaton etn_time_t format to UNIX timeval format. The formal definition of `etn_ETNtoUNIXtime64()` is:

```c
/**
 * @param nETN uint64_t etn_time_t value containing valid time data.
 * @param pUnix Pointer to timeval struct - will contain converted data.
 * @return A non-zero return value indicates an error condition.
 */
int etn_ETNtoUNIXtime64(uint64_t nETN, timeval *pUnix);
```

**4.1.10 etn_ETNtoETNtime64()**

Converts Eaton etn_time_t format to a uint64_t value. The formal definition of `etn_ETNtoETNtime64()` is:

```c
/**
 * @param pETN Pointer to etn_time_t struct containing valid time data.
 * @return ETN Time as a uint64_t value.
 * A zero return value indicates an error condition.
 */
uint64_t etn_ETNtoETNtime64(etn_time_t *pETN);
```
4.2 Time zone Functions

4.2.1 *etn_GetLocalTimeZone*

Returns the difference in seconds between coordinated universal time (UTC) and local time. The formal definition of *etn_GetLocalTimeZone* is:

```c
/**
 * @return Number of seconds west of GMT.
 */
long etn_GetLocalTimeZone();
```

4.2.2 *etn_GetLocalTimeZoneName*

Returns the Timezone and DST names. The formal definition of *etn_GetLocalTimeZoneName* is:

```c
/**
 * @return char ** pointer to the tzname[] vector. Where:
 *    * tzname[0] is the timezone name derived from TZ
 *    * environment variable.
 *    * tzname[1] is the DST zone name derived from TZ
 *    * environment variable. If DST zone is omitted from
 *    * TZ, tzname[1] is an empty string.
 */
char ** etn_GetLocalTimeZoneName();
```

4.2.3 *etn_IsDaylightSavings*

Returns a Boolean value indicating if Daylights Savings is active. The formal definition of *etn_IsDaylightSavings* is:

```c
bool etn_IsDaylightSavings();
```
4.3 Trace Functions

4.3.1 etn_NTPtrace()
Determines where a given Network Time Protocol (NTP) server gets its time from, and follows the chain of
NTP servers back to their master time source. This function is based on the NTPQ code from the NTP
project. The formal definition of etn_NTPtrace() and related structures are:

```cpp
/* Structure for storing NTP trace information */
typedef struct _ntptrace_t
{
    std::string ipaddr; /* IP address of the NTP server. */
    std::string hostname; /* IP/hostname of the NTP server. */
    std::string refid; /* IP address of the reference NTP server for this server. */
    std::string system; /* System type running NTP. */
    bool timeout; /* Indicates whether a timeout occurred for this trace. */
    int stratum; /* NTP stratum level. */
    float offset; /* Clock reference offset. */
    float rootdelay; /* */
    float rootdispersion; /* */
    float syncdistance; /* Host synchronization distance */
} ntptrace_t;

/* Define a vector type for NTP trace information */
typedef std::vector<ntptrace_t> ntptrace_vector_t;

/**
 * Builds a ntptrace_vector_t vector of ntptrace_t structures.
 * @param ntpvector ntptrace_vector_t vector to populate.
 * @param hostname IP/hostname address to trace.
 * @return Number of NTP traces traversed.  -1 if failure has occurred.
 */
int etn_NTPtrace(ntptrace_vector_t &ntpvector, char *hostname, uint32_t timeout);
```

4.3.2 etn_IPtrace()
Traces a network route between two systems, listing all the intermediate routers a connection must pass
through to get to its destination. The Win32 version of this function is based on the Komodia library, and
the Linux version on the UNIX traceroute code. The formal definition of etn_IPtrace() and related
structures are:

```cpp
/* Structure for storing IP trace information */
typedef struct _iptrace_t
{
    std::string ipaddr; /* IP address of the hop point. */
    std::string hostname; /* Hostname of the hop point. */
    bool timeout; /* Indicates whether a timeout occurred for this trace. */
    uint8_t hop; /* TTL hop level. */
    uint32_t elapsedtime; /* Elapsed time between the packet transmission and the */
                        /* reception of the corresponding ICMP packet. */
} iptrace_t;

/* Define a vector type for IP trace information */
typedef std::list<iptrace_t> iptrace_vector_t;

/**
 * Builds a iptrace_vector_t vector of iptrace_t structures.
 * @param ipvector iptrace_vector_t vector to populate.
 * @param hostname IP/hostname to trace.
 * @param port TCP/IP port trace will use to send queries.
 * @param timeout Timeout period (in milliSeconds) for responses to trace queries.
 * @return Number of IP traces traversed.  -1 if failure has occurred.
 */
int etn_IPtrace(iptrace_vector_t &ipvector, char *hostname, uint32_t timeout, int port, uint32_t timeout);
```
int etn_IPtrace(iptrace_vector_t &ipvector, char *hostname, uint16_t port, uint32_t timeout, uint8_t maxhops);
5 Time Zone and Daylight Savings

Time zone and Daylight Savings management will be handled via Web page interface and CGI scripting on the embedded Toolkit system.
6 NTPD Usage Notes for the PXG ARM 9 Linux System.

6.1 Kernel Clock Tick

Initial testing with the ARM NTPD daemon showed that it was not synching properly with NTP servers. After the initial system clock set to the correct time, NTP would periodically reset the system clock in 1-second increments. The fault lay in that the default kernel clock tick value was advancing the system clock too fast for the NTP clock loop algorithm to control. The NTP utility program `tickadj` showed the clock tick at 10009. By adjusting the clock tick with `tickadj` (NTPD was restarted after every tick adjustment), and viewing the how fast the NTPD offset and jitter data were changing with the `ntpq` program, it was determined that a tick value of 10000 allowed NTPD to sync with NTP servers, and control the system clock properly to maintain synchronization. The Linux system init script `/etc/init.d/rcS` was modified with the following lines to set the kernel clock tick at system boot up:

```bash
# Adjust the system clock tick appropriately so that NTP can work
[ -x "/bin/tickadj" ] && tickadj 10000
```

6.2 NTPD Panic Threshold Test is Always Disabled

The `-g` option is supposed to allow NTP to set the system clock ahead or behind a large amount without throwing a panic threshold error, and exiting the program. However, testing showed that this was not always the case, especially when using the Local Clock options “server 127.127.1.0” and “fudge 127.127.1.0 stratum 10” in the NTP.CONF file. In some cases, NTPD would generate the system error message “time correction of X.Y seconds exceeds sanity limit; set clock manually to the correct UTC time”; and the program would exit.

To prevent this from occurring in embedded applications, the NTPD panic threshold test has been disabled in the Toolkit version of the NTP daemon.

6.3 NTPD Default Start-up Options

The Linux system NTPD init script `/etc/init.d/ntpd` uses the `-g` and `-c` options for starting NTPD (example below). The `-g` option is called even though its functionality is hard coded to always be on in the Toolkit version of NTPD (see 6.2). The `-c` option sets the NTP configuration file name and path (if different from default).

```bash
ntpd -g -c /etc/ntp.conf
```

6.4 Extended Server Options in the NTP.CONF file

Testing with NTP server polling options showed that setting the server options `minpoll` to 4, and `maxpoll` to 6 gave the best balance between quickly syncing up to NTP servers, but avoids over polling NTP servers in the manner the `burst` option does. A command-line example of setting an NTP server value with `minpoll` and `maxpoll` values in the NTP configuration file is shown below:

```bash
ntpd -c /etc/ntp.conf -S "192.168.1.102 minpoll 4 maxpoll 6" -E ""
```
6.5  Subscribing to a Change in NTP Status

The C++ code below shows an example of registering to, and testing the CDS event queue for NTP status change events.

```cpp
cds::CCds    cdsApi;
api_status_t retcode;
std::string  basePath;

basePath = SWPROC_NTP + SW_STATUS + "/Value";

if (!cdsApi.init()){ /* Register for status change events from NTPD */
    cdsApi.event_register(basePath),(api_event_t)(EVENT_NOTIFY), retcode);
/* Process events in the queue */
while ( cdsApi.event_pending() ) { int rval = cdsApi.events_get(events);
    if ( rval > 0 ) { assert(static_cast<size_t>(rval) == events.size());
        for (size_t k=0; k<events.size(); k++) {
            if (!events[k].path.empty()) { /* Check for NTP status event */
                if ( event.path.find(basePath) == 0 ) {
                    /* Do something here */
                }
            }
        }
    }
}
}
```

6.6  Subscribing to a NTP Clock Step Event

The C++ code below shows an example of registering to, and testing the CDS event queue for NTP clock step events.

```cpp
cds::CCds    cdsApi;
api_status_t retcode;

if (!cdsApi.init()){ /* Register for clock step events from NTPD */
    cdsApi.event_register(VALUE_LEAF(NTP_CLOCKSTEP),(api_event_t)(EVENT_CHANGE), retcode);
/* Process events in the queue */
while ( cdsApi.event_pending() ) { int rval = cdsApi.events_get(events);
    if ( rval > 0 ) { assert(static_cast<size_t>(rval) == events.size());
        for (size_t k=0; k<events.size(); k++) {
            if (!events[k].path.empty()) { /* Check for NTP clock step event */
                if ( event.path.find(VALUE_LEAF(NTP_CLOCKSTEP)) == 0 ) {
                    /* Do something here */
                }
            }
        }
    }
}
```
7 EULA and Copyright Notices of 3rd Party Code Sources

Source code from the following 3rd party projects has been incorporated into the Linux and Windows versions of libtime. These projects use a BSD style license, which requires copyright acknowledgement, but does not require altered source code to be published, or produce on demand (i.e. GPL).

7.1 NTP

The following copyright notice applies to all files collectively called the Network Time Protocol Version 4 Distribution. Unless specifically declared otherwise in an individual file, this notice applies as if the text was explicitly included in the file.

******************************************************************************
*                                                                       *
* Copyright (c) David L. Mills 1992-2006                                *
*                                                                       *
* Permission to use, copy, modify, and distribute this software and    *
* its documentation for any purpose and without fee is hereby granted, *
* provided that the above copyright notice appears in all copies and   *
* that both the copyright notice and this permission notice appear in  *
* supporting documentation, and that the name University of Delaware    *
* not be used in advertising or publicity pertaining to distribution of *
* the software without specific written permission. The University of   *
* Delaware makes no representations about the suitability this software  *
* for any purpose. It is provided "as is" without express or implied    *
* warranty.                                                            *
******************************************************************************

7.2 Traceroute

Copyright (c) 1988 Regents of the University of California.
All rights reserved.

Redistribution and use in source and binary forms are permitted provided that the above copyright notice and this paragraph are duplicated in all such forms and that any documentation, advertising materials, and other materials related to such distribution and use acknowledge that the software was developed by the University of California, Berkeley. The name of the University may not be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED ``AS IS'' AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

PSC Changes Copyright (c) 1992 Pittsburgh Supercomputing Center.
All rights reserved.

Redistribution and use in source and binary forms are permitted provided that the above copyright notice and this paragraph are duplicated in all such forms and that any documentation, advertising materials, and other materials related to such distribution and use acknowledge that the modifications to this software were developed by the Pittsburgh Supercomputing Center. The name of the Center may not be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED ``AS IS'' AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
7.3 Komodia

Copyright (c) 2000-2003 Barak Weichselbaum <barak@komodia.com>
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
8 References and Links

8.1 RFC-1305

8.2 NTP.Org
www.ntp.org - Home of the Network Time Protocol project.

8.3 NTP Public Service Project
http://ntp.isc.org/bin/view/Main/WebHome - Home of the NTP Public Services Project.

8.4 Time and Frequency Software

8.5 Windows NTP Software