

TRAX: DTN RealTime

Programming Interfaces and Data Capture for DTN RealTime

**Technical Guide
Document Version 1.16**

Base 16 Corporation

Base 16 Corporation licenses the use of TRAX: DTN RealTime for use with DTN RealTime (Data Transmission Network) market data services. Base 16 Corporation does not warrant, represent or guarantee the accuracy, suitability or merchantability of the software or raw information presented. Base 16 Corporation strives to provide software that is reliable and without defects. In no manner may Base 16 Corporation be held liable for any defect, flaw or error in data or information when using this product. DTN nor DTN Market Access are not liable for any defect, flaw or error in data or the information provided.

THE SOFTWARE AND INFORMATION IS PROVIDED TO THE USERS "AS IS." BASE 16 CORPORATION MAKES NO EXPRESS OR IMPLIED WARRANTIES OF ANY KIND REGARDING THE SOFTWARE OR INFORMATION, INCLUDING, WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE OR USE. BASE 16 CORPORATION WILL NOT BE LIABLE TO ANY USER OR ANYONE ELSE FOR ANY INTERRUPTION, INACCURACY, ERROR OR OMISSION, REGARDLESS OF CAUSE, IN THE INFORMATION OR FOR ANY DAMAGES (WHETHER DIRECT OR INDIRECT, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY) RESULTING THEREFROM.

Copyright 2004 Base 16 Corporation
All rights reserved.
November 8, 2004

Base 16 Corporation
1802 Pleasant Valley Dr., Suite 100-360
Garland, TX 75040
(972) 463-3080

<http://trax.base16.com>

Table of Contents

1	Overview	1
2	Installation	2
2.1	Connecting to the DTN Receiver	2
2.2	Continuous Operation.....	3
2.3	Configuration	4
2.4	[dtn-receiver] Section	4
2.5	[service] Section.....	4
2.6	[database-n] Section.....	6
3	Control Program	8
3.1	Receiver	8
3.1.1	Controlling the Service	8
3.1.2	DTN Receiver	8
3.1.3	Authorizations	9
3.1.4	Refreshing Records	9
3.2	Statistics	9
3.3	Sockets	10
4	Socket Interface	11
4.1	Connecting and Registering for Events.....	11
4.2	Testing with Telnet	11
4.3	Commands	11
4.3.1	TRADES.....	12
4.3.2	QUOTES	12
4.3.3	FIELDS.....	12
4.3.4	MKTCOND.....	12
4.3.5	BARS	12
4.3.6	ALL	13
4.3.7	GETRECORD	13
4.3.8	REFRESHRECORD	13
4.3.9	REFRESHALLRECORDS	13
4.3.10	GETOPTIONCHAIN	14
4.3.11	NAME.....	14
4.3.12	HELP.....	14
5	Event Messages	15
5.1	Data Types	15
5.2	Trade Event	16
5.3	Quote Bid/Ask Event	17
5.4	Quote Event.....	17
5.5	Market Condition Event	18
5.6	Field Update Event.....	18
5.6.1	Field Names.....	19
5.7	Bar Event	20
5.8	Record Event.....	21
5.9	Option Chain Event	22
5.10	Service Status Event	23
6	Capture Files	24
6.1	System Resource Demands.....	24
6.2	Using Excel and Other Applications with Capture Files	24
6.3	Directory Structure	24
6.3.1	Bars Capture.....	25
6.3.2	Tick Capture	25
6.4	File Structure	25
6.4.1	Bar Files.....	25
7	Shared Memory Interface	26
7.1	Shared Memory Fields	27
7.2	Accessing Shared Memory	31
7.3	Special Status Record.....	31
7.4	Scanning Market Data	32
8	System Files	33
8.1	Symbol Files	33
8.2	Options Files	33
8.3	Market Identifiers	33
8.4	Market Conditions	34
8.5	Field Names	34

8.6	Shared Memory Descriptions	34
8.7	Log Files	34
9	COM Socket Interface	36
9.1	Connect method	36
9.2	Application Name property	37
9.3	Address property	37
9.4	Port property	37
9.5	Connection State property	37
9.6	Disconnect method	37
9.7	SendCommand method	37
9.8	OnReceive event	37
9.9	OnConnection event	37
10	COM Shared Memory Interface	38
10.1	Add method	38
10.2	Find method	38
11	RTD Server	39
11.1	Security	39
11.2	Refresh Interval	40
11.3	RTD Fields	41
12	Wealth-Lab Adapter	42
12.1	Configuration File	42
13	Appendix A	43
14	Appendix B - Receiver and Refreshing Record Data	45
15	Appendix C - DTN Supported Databases	46
16	Appendix D - Correction Indicator	47

1 Overview

TRAX: DTN RealTime is an add-on software package providing programming interfaces and automatic data capture for DTN's RealTime satellite market data services.

For Software Developers

- Access DTN RealTime data through easy to use programming interfaces
- Reduce development effort
- Program applications in C++, C, Visual Basic, Delphi, Java and other languages
- Scan the entire market in real time
- Identify unique trading opportunities
- Capture market data for back testing and charting
- Monitor an unlimited number of symbols

For Non-Programmers

Access real time data directly from Microsoft Excel 2002 spreadsheets

TRAX: DTN RealTime's core module runs in the background as a service on Windows 2000 or Windows XP. The service handles the direct interface to the DTN receiver, maintains an in-memory real time database and can automatically capture market data. The following interfaces are provided to easily access the data.

Socket Interface - Receive real time streaming market data over TCP/IP sockets
Shared Memory Interface - Access real time data in shared memory
Files - Directly read data files of captured bars and ticks

COM Interface - For Visual Basic and other COM supported languages
RTD Server - Access data directly from Microsoft Excel 2002 spreadsheets

Supports stocks, indexes, equity options and futures.

2 Installation

To install TRAX: DTN RealTime, follow the directions given in the INSTALL-README.txt file included with the distribution.

After installing the software on your system, you'll need to perform the following steps:

Step 1 - Get receiver settings for the DTN receiver using Chameleon software supplied by DTN. See section 2.1.

Step 2 - Edit the configuration file, *dtn-realtime.conf* and modify the receiver, database and license sections. See section 2.3.

Step 3 - Create and start the core service using the control program. See section 3.1.1

Step 4 - Use telnet and the included test programs to verify operation of the socket interfaces and the memory interface. See section 4.1

2.1 Connecting to the DTN Receiver

TRAX: DTN RealTime requires an Ethernet connection to the DTN receiver. It is recommended that a dedicated Ethernet adapter is used for the connection since a large volume of data is transmitted over the connection. The connection may be through a hub or Ethernet switch, or may be connected directly using a cross-over cable.

First, setup the DTN receiver and be sure that it functions properly with the DTN supplied Chameleon software. Follow the instructions included with Chameleon. If you have any questions about installation of the receiver or the use of the Chameleon software, please contact DTN Customer service.

Once you're sure that data is being received by Chameleon, you will need information about the receiver port settings. Start Chameleon Release 3. Select **Configuraton**. Select the **Data Settings** tab. Make a note of the settings for D8000 IP Address, D8000 TCP Port and Client UDP Port in the table below. Close the Chameleon program. TRAX: DTN RealTime cannot connect to the DTN receiver at the same time as Chameleon.

D8000 IP Address	
D8000 TCP Port	
Client UDP Port	

Edit the TRAX: DTN RealTime configuration file *dtn-realtime.conf* (described in the next section). Set the parameters from the Chameleon settings. For instance if the Chameleon parameters are:

D8000 IP Address	10.100.116.110
D8000 TCP Port	21560
Client UDP Port	20547

the *[dtn-receiver]* section of the configuration file should look like this:

```
ReceiverAddress = 10.100.116.110
TCPPort        = 21560
UDPPort        = 20547
```

2.2 Continuous Operation

To be most effective TRAX: DTN RealTime should run on a computer that is powered up and connected to the DTN receiver 24 hours a day. This allows TRAX to maintain the in-memory database to reflect the most current market data and allows your applications to readily access all available market data.

However, it is also possible to start TRAX before or during the market day. When TRAX is first started, it will begin to receive market data from the DTN receiver and will begin to populate the TRAX in-memory database.

TRAX will automatically refresh records for symbols that it receives data updates for, but will not refresh records for securities that are not active (no quotes or trades, etc.) thus thinly traded securities may not be current in the TRAX in-memory database.

To cause the in-memory database to obtain the complete set of current data from the receiver, you may issue REFRESHRECORD or REFRESHALLRECORDS commands from a client socket or initiate Refresh Records or Refresh All Records requests from the TRAX: DTN RealTime control program. Refreshing all records may take up to 5 minutes depending on the data services that you're subscribed to. During this time, TRAX: DTN RealTime can be used as normal, albeit, data will become available as it is received.

If TRAX: DTN RealTime is stopped or the computer is rebooted, the in-memory data is lost, and the user should initiate Refresh All Records in one of the methods described above.

Please see Appendix B - Receiver and Refreshing Record Data for more information.

2.3 Configuration

TRAX: DTN RealTime uses a configuration file to control various features. The file, ***dtn-realtime.conf*** must be located in the same directory as *DTNRealTime.exe*. Use an ordinary ASCII text editor such as NotePad to edit the file. For new configuration changes to take effect, the TRAX: DTN RealTime core service must be stopped and restarted. See section 3.1.1 for details.

The hash-symbol # is the comment character, everything from the hash mark to the end of the line is considered a comment.

2.4 [dtn-receiver] Section

The *dtn-receiver* section contains the connection parameters to the DTN D80x0 receiver.

Parameter	Description
ReceiverAddress	The IP address of the DTN receiver
TCPPort	The TCP port on the DTN receiver. Default is 21560.
UDPPort	The UDP port on the DTN receiver. Default is 20547.
TCPUpdates	If set to TRUE, TRAX: DTN RealTime will connect to the receiver using TCP to collect real time updates. If you do not have a D8080 receiver this will be ignored since a D8080 receiver is required to receive using TCP.

Example:

```
[dtn-receiver]
ReceiverAddress = 10.100.116.110
TCPPort        = 21560
UDPPort        = 20547
TCPUpdates     = TRUE
```

2.5 [service] Section

The *service* section contains the parameters that control the core TRAX: DTN RealTime service.

Parameter	Description
ClientPort	TCP port that applications connect to, i.e., the Socket Interface
ControlPort	TCP control port used for control and monitoring purposes.
MaxClients	Maximum number of clients (via Socket Interface) that are allowed to connect.
MaxControls	Maximum number of control applications that are allowed to connect.
DailyRestartTime	The service automatically stops and restarts at this time. This process is required to ensure that the service can receive the latest real-time information each day. Set values in 24 hour format, i.e., for 7:00pm use 19:00. Default : 00:00 (midnight local time).
AutoRefreshTime	The service will automatically initiate Refresh All Records at this time. See section 3.1.4 and section 14 for details on Refreshing Records. The time specified must be after the DailyRestartTime or else it will be ignored. Also, be sure to allow adequate time for all of the records to refresh before the OpenClearTime (see below). Set values in 24 hour format, i.e., for 7:00pm use 19:00. If this line is not present in the file or is commented out, no auto refresh will be performed. Default : 04:00
OpenClearTime	The service automatically clears fields in the in-memory database at this time. The following fields will be cleared:

	Open, High, Low, Last, Bid, BidSize, BidTime, Ask, AskSize, AskTime. The time specified must be after the DailyRestartTime or else it will be ignored. If this line is not present in the file or is commented out, no clearing will be performed. Set values in 24 hour format, i.e., for 7:00pm use 19:00. Default : 05:30
LogFilePath	Directory where the daily log file will be written
FilePath	Directory used for files downloaded from the DTN receiver and other system files. See section 8 for details.
ListPath	Base directory for user symbol lists.
EventBuffers	Number of event buffers. Default : 2000. Can be adjusted higher if "Event receiver overflow" errors occur (appearing in the <i>dtn-realtime-log.txt</i> log file) regularly indicating that TRAX ran out of event buffers. Maximum is 10000.

Example:

```
[service]
ClientPort      = 8700
ControlPort     = 9001
MaxClients      = 16
MaxControls     = 5
LogFilePath     = C:\Program Files\Trax\bin\log
FilePath        = C:\Program Files\Trax\bin\files
ListPath        = C:\Program Files\Trax\bin\list
```

2.6 [database-n] Section

Market data provided by DTN is organized and referred to as databases. If you're subscribed to DTN RealTime equities, you will be authorized to receive data for the following databases:

NYSE STOCKS
 NASDAQ STOCKS
 AMEX STOCKS
 INDEXES
 DTN CALCULATED INDICATORS

If you also subscribe to equity options, you will also be authorized for:

REAL TIME STOCK OPTIONS

If you subscribe to CME futures you'll be authorized for:

MERC FUTURES

See Appendix C - DTN Supported Databases for a complete listing of databases.

The **database-n** sections contain parameters that control how data is handled by TRAX for each database authorized for your DTN subscription. Each database section in the configuration file must contain a section heading [database-n] where "n" is a unique number ranging from 0-29, for instance [database-0]. It is not important if the sections are in numerical or consecutive order.



Capturing data can consume significant computer resources such as processor time and disk space. Please read the Data Capture section before enabling capture.

Parameter	Description
Name	The DTN assigned name. See Appendix C - DTN Supported Databases for a complete listing of databases.
CaptureList	A file containing a list of symbols (one symbol per line) that should be captured. Remove or comment this line out with a # character to capture data for all symbols.
Capture	Specify what to capture. Use a space between several values. BARS - Capture minute bars TRADES - Capture trades QUOTES - Capture quotes If no settings are present no data will be captured for this database.
CaptureStart	The time to start capturing bars and tick data. Set this time and CaptureEnd to limit the time of day to capture data. Set values in 24 hour format. i.e., for 7:00pm use 19:00. Default is 5:00.
CaptureEnd	The time to end capturing bars and tick data. Set this time and CaptureStart to limit the time of day to capture data. Set values in 24 hour format. i.e., for 7:00pm use 19:00. Default is 21:00.
CaptureDir	The directory where the capture files will be written.
SharedMemory	Specify which symbols to maintain in shared memory. ALL - All symbols of the database OFF - No symbols will be maintained in shared memory for this database.
SharedMemoryFile	The name of the shared memory backing file. Several databases may specify the same SharedMemoryFile, for instance to place NYSE STOCKS, NASDAQ STOCKS and AMEX STOCKS in the same shared memory area. Databases must be of the same type. Base Type : STOCKS, INDEXES, DTN CALCULATED INDICATORS Options Type: REAL TIME STOCK OPTIONS Futures Type: FUTURES
SharedMemoryExt	Set to TRUE to use the shared memory extensions instead of standard (base) shared memory records.

	<p>For Equity Databases (NYSE, NASDAQ and AMEX): Allows equity (stock) fundamental data fields to be accessible through shared memory but will increase the size of the shared memory area by approximately 2.5MB.</p> <p>When <i>SharedMemoryExt</i> is set to TRUE, use the <i>TraxDtnRtEquityExtMemoryRecord</i> structure to access shared memory records instead of the structure <i>TraxDtnRtMemoryRecord</i>.</p> <p>Set to FALSE to save memory if your application doesn't require shared memory access to equity fundamental fields or to maintain compatibility. Fundamental fields can also be accessed over the Socket Interface by monitoring for Field Update events.</p> <p>It is important to set each [database-n] section for each database using the same <i>SharedMemoryFile</i> to identical settings for <i>SharedMemoryExt</i>. In other words, if there are 3 databases, NASDAQ STOCKS, NYSE STOCKS and AMEX STOCKS using the same <i>SharedMemoryFile</i>, each must contain the parameter <i>SharedMemoryExt</i> and be set to the same value in each.</p> <p>For REAL TIME STOCK OPTIONS Database: Allows one additional option record field - PreviousVolume - to be accessed through shared memory.</p> <p>When <i>SharedMemoryExt</i> is set to TRUE, use the <i>TraxDtnRtOptionExtMemoryRecord</i> structure to access shared memory records instead of the structure <i>TraxDtnRtOptionMemoryRecord</i>.</p> <p>If this parameter is not specified in the configuration file, it will default to FALSE.</p>
--	--

Examples:

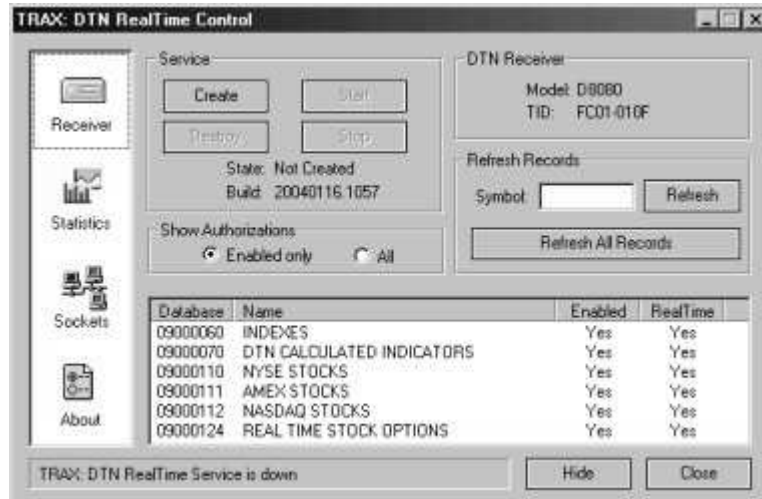
```
[database-0]
Name           = NASDAQ STOCKS
Capture        = BARS TRADES QUOTES
CaptureStart   = 7:00
CaptureEnd     = 19:30
CaptureDir     = C:\Program Files\Trax\data\stock
SharedMemory   = ALL
SharedMemoryFile = C:\Program Files\Trax\memory\stock
SharedMemoryExt = TRUE

[database-1]
Name           = REAL TIME STOCK OPTIONS
Capture        = TRADES QUOTES BARS
CaptureStart   = 7:00
CaptureEnd     = 19:30
CaptureDir     = C:\Program Files\Trax\data\option
SharedMemory   = ALL
SharedMemoryFile = C:\Program Files\Trax\memory\option
SharedMemoryExt = TRUE

[database-2]
Name           = MERC FUTURES
Capture        = TRADES QUOTES BARS
CaptureStart   = 7:00
CaptureEnd     = 19:30
CaptureDir     = C:\Program Files\Trax\data\future
SharedMemory   = ALL
SharedMemoryFile = C:\Program Files\Trax\memory\future
```

3 Control Program

The TRAX: DTN RealTime Control Program is used to control and monitor the core service. The Control Program can be closed and opened as needed without affecting the TRAX core service. The Control panel can be hidden if desired, and reactivated from a task bar icon. The task bar icon reflects the operational status of the core service, blue when the service is OK, flashing red if a problem occurs.



Clicking on the Hide button or the minimize button will hide the window. To reactivate or show the window, click on the TRAX task bar icon in the task bar tray area at the lower right section of your taskbar. Clicking on the menu page items in the left area will select various pages of the program.

3.1 Receiver

The receiver page contains controls to conveniently create, destroy, start and stop the core service. Service status, information about the DTN receiver and controls to refresh records are also included.

3.1.1 Controlling the Service

Pressing the Create button will create the TRAX: DTN RealTime core as a self-starting Windows service. DTNRealTime will then be listed in the "Services" control panel of Windows, as a service automatically started up at boot time. The DTNRealTime service runs in the background, so you will not see it on your desktop. Pressing the Create button creates (installs) the service and automatically starts it.



To create and start the service, Press **Create**.

Pressing the Destroy button will stop the service, and remove it from registration with the Windows Service Manager. When a service is registered with the Service Manager, it can be controlled from the Windows Services interface and can automatically start at Windows boot time.

Pressing Start or Stop will start or stop the service, but leave it registered with the Service Manager.

3.1.2 DTN Receiver

This section displays the Model and the Terminal ID (TID) of the attached DTN receiver.

3.1.3 Authorizations

The lower section of the Receiver area contains a list of authorizations. These are the DTN databases (data services) that your receiver is authorized to receive and depend on your subscription with DTN. The Show Authorizations buttons allow you to view all authorizations or only those which you have enabled in the *dtn-realtime.conf* configuration file.

The list contains the DTN database identifier in hex format, the name of the database, whether the database is enabled and whether it is real time or delayed data. If the database is enabled in the *dtn-realtime.conf* configuration file, the Enabled column will display "Yes", otherwise it will be blank. Likewise, the RealTime column shows if the database is real time.

3.1.4 Refreshing Records

The Refresh Records section provides controls to request record refresh for one or all symbols. The DTN satellite receiver (D80x0) contains the latest market data. The TRAX: DTN RealTime service is designed to run 24 hours a day. Each night the DTN receiver broadcasts an automatic refresh record update cycle. This update is collected by TRAX: DTN RealTime and ensures that the in-memory database is up-to-date. The *AutoRefreshTime* in the configuration file can also initiate an automatic refresh of all records at a specific time and is recommended.

Should TRAX: DTN RealTime be stopped or the computer rebooted, the in-memory data will be lost. To restore the in-memory data for a particular symbol, enter the symbol and press the Enter key, or click Refresh. Click the Refresh All Records button to restore the in-memory data for all enabled databases. This will take several minutes, but is recommended if the service was not running during the normal overnight refresh cycle.

Applications may also issue REFRESHRECORD commands through the Socket Interface. Refreshing records, either with the control program, or with the Socket Interface will generate Field Update and Market Condition events for each record.

3.2 Statistics

The statistics pane shows a display of the received bytes, packets, errors and other statistics related to communication with the DTN receiver.

The Event Port section contains statistics related to real time update events received from the DTN receiver.

Statistic	Description
RxBytes	The number of bytes received
RxPackets	The number of packets received
RxErrors	The number of errors received. When using TCP with D8080 receivers, this should be zero. D8000 receivers must use UDP and some errors are normal.
Queue Max	The maximum number of packets queued internally.
Refreshes	The number of record refreshes received.
Refresh Queue	The number of refresh requests queued. After requesting "Refresh All Records" the queue will be filled and decrease down to zero as requests complete.
Updates	The number of field updates received.
Trades	The number of trades received.
Quotes	The total number of quotes received (includes both bids and asks)
Bid Quotes	The number of bid quotes received.
Ask Quotes	The number of ask quotes received.

The Request Port section contains statistics related to the request port used with the DTN receiver.

Statistic	Description
RxBytes	The number of bytes received
RxPackets	The number of packets received
RxErrors	The number of errors that occurred while receiving.
TxBytes	The number of bytes sent.
TxPackets	The number of packets sent
TxErrors	The number of errors that occurred while sending.

3.3 Sockets

The Sockets pane contains statistics pertaining to application sockets.

Statistic	Description
Name	The name of the application. Applications set the name by sending the NAME command.
RxBytes	The number of bytes received
RxErrors	The number of errors that occurred while receiving
TxBytes	The number of bytes received
TxErrors	The number of errors that occurred while sending
TxBuffPk	The peak number of bytes that were buffered. The maximum is 2,097,152 (2MB). If this limit is exceeded, data will be lost.

4 Socket Interface

The Socket Interface allows applications to receive streaming real time trades, quotes and other market data.

4.1 Connecting and Registering for Events

Follow these steps to receive data over the Socket Interface:

- 1) Connect to the client TCP socket. Use the local host loopback address 127.0.0.1 and the ClientPort designated in *dtn-realtime.conf*. The default port is 8700.
- 2) Issue one or more commands to register for events or request data over the connection.
- 3) Begin receiving data events

A COM component is included with TRAX that can be used with Visual Basic and other COM-aware languages to send and receive data over the socket interface. See section 9 for details.

4.2 Testing with Telnet

Normally, programs will connect to the client socket, however, during testing it is sometimes useful to use a Telnet application to connect and receive data. To use Telnet, from the Windows Start menu, select Run..., then type in:

```
telnet 127.0.0.1 8700
```

and press OK.

When the telnet window starts, type HELP <enter>. To begin receiving quotes for MSFT, type TRADES,ON,MSFT <enter>.

You may use backspacing to edit your commands. If a command is not valid, no response will be received. Be careful when using the ALL command, as the Telnet application is not able to keep up with the volume of data that will be received.

4.3 Commands

Commands are ASCII comma-delimited strings terminated with a newline (\n or 0x0A or \bLf) character. For example to register for trade events for CSCO, send the following string:

```
TRADES,ON,CSCO
```

To stop receiving events for CSCO trades, send:

```
TRADES,OFF,CSCO
```

Many commands require a symbol as a parameter. When referring to an option symbol, use a leading period to indicate that the symbol is an option symbol.

```
TRADES,OFF,.QQQNL
```

Many of these commands can also handle more than one symbol, for instance:

```
REFRESHRECORD,CSCO,MSFT,IBM
```

The command buffer is limited to 2048 characters, so you may list as many symbols as you wish, not to exceed the buffer limit. Use multiple commands for more symbols.

4.3.1 TRADES

Syntax: TRADES,[ON | OFF],<symbol>,<symbol>,...

Register to receive or stop receiving trade events for one or more symbols.

To register for all symbols: TRADES,ON

To register for several symbols, use: TRADES,ON,CSCO,MSFT,IBM

Use TRADES,OFF in a similar fashion to stop receiving trade events.

4.3.2 QUOTES

Syntax: QUOTES,[ON | OFF],<symbol>,<symbol>,...

Register to receive or stop receiving quote events (Quote Bid/Ask Events and Quote Events) for one or more symbols.

To register for all symbols: QUOTES,ON

To register for several symbols, use: QUOTES,ON,CSCO,MSFT,IBM

Use QUOTES,OFF in a similar fashion to stop receiving quote events.

4.3.3 FIELDS

Syntax: FIELDS,[ON | OFF],<symbol>,<symbol>,...

Register to receive or stop receiving field events for one or more symbols.

To register for all symbols: FIELDS,ON

To register for several symbols, use: FIELDS,ON,CSCO,MSFT,IBM

Use FIELDS,OFF in a similar fashion to stop receiving field events.

4.3.4 MKTCOND

Syntax: MKTCOND,[ON | OFF],<symbol>,<symbol>,...

Register to receive or stop receiving Market Condition events for one or more symbols. Market Condition events occur to alert a condition such as a Trading Halt, Crossed Market, Locked Market, etc.

To register for all symbols,: MKTCOND,ON

To register for several symbols, use: MKTCOND,ON,CSCO,MSFT,IBM

Use STATUS,OFF in a similar fashion to stop receiving status events.

4.3.5 BARS

Syntax: BARS,[ON | OFF],<symbol>,<symbol>,...

Register to receive or stop receiving minute bar events for one or more symbols.

You must explicitly designate each symbol that you would like to receive bars for. You cannot register for all symbols for bars as you can in many other commands

To register for a symbol, use: BARS,ON,CSCO

To register for several symbols, use: BARS,ON,CSCO,MSFT,IBM

Use BARS,OFF in a similar fashion to stop receiving bar events.

4.3.6 ALL

Syntax: ALL,[ON | OFF],<symbol>,<symbol>,...

Register to receive or stop receiving all events, except bars, for one or more symbols.

To register for all symbols: ALL,ON

To register for several symbols, use: ALL,ON,CSCO,MSFT,IBM

Use ALL,OFF in a similar fashion to stop receiving events.

BARS must be registered with the BARS command and each symbol must be designated.

4.3.7 GETRECORD

Syntax: GETRECORD,<symbol>,<symbol>,...

Request records for one or more symbols. If the symbol exists in the in-memory database, a Record Event will be sent. See the Record Event section for details. This command is useful to initialize an application with the most current receiver record data. Also see Appendix B - Receiver and Refreshing Record Data for a diagram of how GETRECORD differs from REFRESHRECORD.

To request a record: GETRECORD,IBM

To request records for several symbols, use: GETRECORD,CSCO,MSFT,IBM

4.3.8 REFRESHRECORD

Syntax: REFRESHRECORD,<symbol>,<symbol>,...
REFRESHRECORD

Request refresh of record data from the DTN receiver for one or more symbols. See Appendix B - Receiver and Refreshing Record Data for a diagram of how REFRESHRECORD differs from GETRECORD.

Refreshing records, either with the control program, or with the REFRESHRECORD command, will generate a Field Update and a Market Condition event for each record.

To request a record refresh: REFRESHRECORD,IBM

To request record refresh for several symbols, use:
REFRESHRECORD,CSCO,MSFT,IBM

4.3.9 REFRESHALLRECORDS

Syntax: REFRESHALLRECORDS

Request refresh of record data from the DTN receiver for all symbols. See Appendix B - Receiver and Refreshing Record Data for more details.

Refreshing records, either with the control program, or with the REFRESHALLRECORDS command, will generate a Field Update and a Market Condition event for each record.

To request a record refresh for all records: REFRESHALLRECORDS

Refreshing records for all symbols will take several minutes. If subscribed to real time stock options, it may take as long as 45 minutes. During this time, TRAX: DTN RealTime can be used as normal, albeit, data will become available as it is received.

4.3.10 GETOPTIONCHAIN

Syntax: GETOPTIONCHAIN,<symbol>
 GETOPTIONCHAINL,<symbol>

Request an option chain for a given symbol. Two forms of the command allow you to request all short term options, or all options including LEAPS. . If options exist for the symbol, a Option Chain Event will be sent. See the Option Chain Event section for details.

To request an option chain excluding LEAPS: GETOPTIONCHAIN,IBM
To request an option chain including LEAPS: GETOPTIONCHAINL,IBM

4.3.11 NAME

Syntax: NAME,<application-name>

Set an application name to identify your application in the socket list. Limit is 50 characters.

The socket list available through the DTN: RealTime control panel or the control interface, lists socket users. The Name column will show the name sent by the application.

Example:

NAME,my-application

4.3.12 HELP

Syntax: HELP

Receive a help menu. Useful for Telnet session users.

When connecting to the service for testing purposes using a Telnet session, this command is useful to list the commands and their syntax. A single question mark (?) may also be used for the help command.

Examples:

HELP
?

5 Event Messages

Event messages are received over the Socket Interface and are composed of ASCII characters in a line-oriented, comma-delimited format. Each line is terminated with a newline character (i.e., \n or 0x0A).

5.1 Data Types

The following data types are used in the fields of the event messages:

Type	Description
Decimal	String of ASCII coded digits. Decimal fields are integers that must be multiplied by a factor (for instance 0.0001) to obtain the true decimal value. The <i>Decimal Code</i> field indicates the multiplier factor for all Decimal fields in a given message.
Float	String of ASCII digits with explicit decimal point (i.e., 21.67).
Integer	String of ASCII coded digits.
Alpha	String of ASCII alphanumeric.
Time	Time in HHMMSS format.
Date	Date in YYYYMMDD format.
DateTime	Date and time in YYYYMMDDHHMMSS format.

The first field of each event message is special in that it is a single alpha string comprised of the event type, the symbol type and the symbol:

Offset	Name	Notes
0	Message Type	Trade Event
1	Symbol Type	Type of symbol code. (See Appendix A)
2	Symbol	Security symbol

Examples:

```
T QQQ
Q .IWOQQ
```

The following table presents a summary of event messages.

Message Type	Notes
A	Ask Quote
B	Bid Quote
C	Option Chain
F	Field Update
M	Market Condition
Q	Quote (Bid and Ask)
R	Bar
S	Record
T	Trade
Z	Status

5.2 Trade Event

The Trade Event indicates that a trade has executed or a correction to previously received Trade Event fields should be made.

	Name	Type	Notes
0	Message Type Symbol Type Symbol	"T"	Trade Event Type of symbol code. (See Appendix A) Security symbol
1	Decimal Code	Integer	Decimal field code (See Appendix A)
2	TradePrice	Decimal	Trade price
3	TradeVolume	Integer	Number of shares or contracts traded
4	TotalVolume	Integer	Total shares traded today
5	Trade Time	Time	Time trade reported HHMMSS
6	Trade Type	Alpha	" " [space] - Regular trade "O" - Opening trade "T" - Form T trade
7	Last Indicator	Alpha	Last indicator (See Appendix A)
8	Market ID	Integer	Market Identifier (See Appendix A)
9	Status	Alpha	Variable status flags. (See Appendix A)

Example:

T QQQ,2,3812,200,33241100,100530,,U,26,*

Note: If the Trade Event contains the Correction indicator "C" in the Status field, the event is a correction, and is **not a new trade**. Applications that "read the tape" may wish to examine the Status field for the Correction indicator and filter out corrections since they do not represent an actual trade. See Appendix D - Correction Indicator for further information on events with the Correction indicator.

5.3 Quote Bid/Ask Event

The Quote Bid/Ask Event contains a bid quote or an ask quote.

	Name	Type	Notes
0	Message Type Symbol Type Symbol	"A"=Ask "B"=Bid	Quote Event Type of symbol code. (See Appendix A) Security symbol
1	Decimal Code	Integer	Decimal field code (See Appendix A)
2	Price	Decimal	Bid/Ask price
3	Size	Integer	Bid/Ask Size
4	Bid Tick Indicator	Alpha	"U" - Up "D" - Down "N" - No Change - (empty) Not indicated
5	Best Bid/Ask Market ID	Integer	Best Bid/Ask Market Identifier (See Appendix A) Not available for options.

Examples:

B NT,2,666,200,,14
A PRU,2,4412,19,,14

5.4 Quote Event

The Quote Event contains a quote containing both a bid and ask quote.

	Name	Type	Notes
0	Message Type Symbol Type Symbol	"Q"	Quote Event Type of symbol code. (See Appendix A) Security symbol
1	Decimal Code	Integer	Decimal field code (See Appendix A)
2	Bid	Decimal	Bid Price
3	BidSize	Integer	Bid Size
4	Ask	Decimal	Ask Price
5	AskSize	Integer	Ask Size
	Bid Tick Indicator	Alpha	"U" - Up "D" - Down "N" - No Change - (empty) Not indicated
7	Best Bid Market ID	Integer	Best Bid Market Identifier (See Appendix A) Not available for options.
8	Best Ask Market ID	Integer	Best Ask Market Identifier (See Appendix A) Not available for options.

Example:

Q PDG,2,1664,39,1665,44,,14,1

5.5 Market Condition Event

The Market Condition Event indicates a change in the market condition. RefreshRecord commands will also cause a Market Condition event.

	Name	Type	Notes
0	Message Type Symbol Type Symbol	"M"	Market Condition Event Type of symbol code. (See Appendix A) Security symbol
1	Market Condition	Numeric	Market Condition Code (See Appendix A)
2	Market ID	Integer	Market Identifier (See Appendix A)

Example: M CSCO,0,19

5.6 Field Update Event

The Field Update Event indicates that one or more fields have been updated. This event is intended to allow an application to keep an external database up-to-date with the latest fundamental market information.

Field ID is (an integer) the index of a field. The names of the fields and the indexes that correspond to them are in the *field-names.csv* located in the FilesPath directory specified in the *dtn-realtime.conf* configuration file. See the following page for a list of fields that are available.

A Field Update event may contain the Correction indicator. See Appendix D - Correction Indicator for further information on events with the Correction indicator.

	Name	Type	Notes
0	Message Type Symbol Type Symbol	"F"	Field Update Event Type of symbol code. (See Appendix A) Security symbol
1	Decimal Code	Integer	Decimal field code (See Appendix A)
2	Status	Alpha	Variable status flags. (See Appendix A)
3	MarketID	Integer	Market Identifier (See Appendix A)
4	Field Count	Integer	Number of fields to follow.
	Field Type	"1" = Integer "2" = Decimal "3" = Float "4" = Time "5" = Date "6" = Time/Date "7" = Alpha	Type of field
	Field ID	Integer	Field Identifier
	Field Data	Depends on Field Type	Data for field
<i>(Variable number of fields)</i>			

Notes:

1) The shaded area shows one field record. The message may contain one or more records.

Examples:

F XOM,2,* ,14,1,1,3,4130

F MSFT,4,* ,19,52,1,5,281900,1,23,281900,1,25,282000,1,12,281300,1,
3,282300,1,4,278500,4,15,125752,1,9,30139773,1,47,31,1,24,373,1,26,
777,1,37,300000,1,38,225500,7,48,-,7,66,MICROSOFT CORP,...

5.6.1 Field Names

These fields will be available from Field Update events.

HIGH
LOW
PE RATIO
52 WEEK HIGH
52 WEEK LOW
XDIV INDICATOR
MARKET IDENTIFIER
ISSUER
XDIV DATE
52 WEEK HIGH DATE
52 WEEK LOW DATE
ANNUAL EST EPS
ANNUAL EPS
PREVIOUS CLOSE
BETA
DIVIDEND YIELD
ANNUAL DIVIDEND
SIC CODE
PREVIOUS VOLUME
PERCENT HELD INSTITUTION
AVERAGE VOLUME
YTD HIGH
YTD HIGH DATE
YTD LOW
YTD LOW DATE
CURRENT QTR DIVIDEND
ASSETS
LIABILITIES
DEBT
DILUTED EPS
FISCAL YEAR END
EPS 3 YEAR GROWTH
EPS 5 YEAR GROWTH
MARKET CAP
DIVIDEND PAY DATE
DIVIDEND RATE
DIVIDEND REC DATE
LAST SPLIT
PREVIOUS SPLIT
LAST SPLIT FACTOR
PREVIOUS SPLIT FACTOR
YEAR END CLOSE
UPC INDICATOR
MARKET CONDITION
TRADE DATE
OPEN INTEREST
STRIKE PRICE
EXPIRATION DATE
SETTLEMENT DATE
CONTRACT HIGH
CONTRACT HIGH DATE
CONTRACT LOW
CONTRACT LOW DATE
OPEN RANGE 1
CLOSE RANGE 1
OPEN RANGE 2
CLOSE RANGE 2

These fields will only be present in full record updates:

OPEN
LAST
LTRADE
FORM T LAST
EXTENDED LAST
TRADE TIME
VOLUME
BID PRICE
BID SIZE
ASK PRICE
ASK SIZE
VOLUME INDICATOR

5.7 Bar Event

The Bar Event occurs when a new minute bar has completed.

	Name	Type	Notes
0	Message Type Symbol Type Symbol	"R"	Bar Event Type of symbol code. (See Appendix A) Security symbol
1	Decimal Code	Integer	Decimal field code (See Appendix A)
2	Open	Decimal	Open Price
3	High	Decimal	High Price
4	Low	Decimal	Low Price
5	Close	Decimal	Close Price
6	Volume	Integer	Volume
7	Start Time	Time	Time bar started HHMM

5.8 Record Event

The Symbol Record Event is sent in response to a GETRECORD,<symbol>,... command.

	Name	Type	Notes
0	Message Type Symbol Type Symbol	"S"	Record Event Type of symbol code. (See Appendix A) Security symbol
1	Decimal Code	Integer	Decimal field code (See Appendix A)
2	Open	Decimal	Open Price
3	High	Decimal	High Price
4	Low	Decimal	Low Price
5	Last Trade	Decimal	Last Trade
6	Last Trade Size	Decimal	Last Trade Size
7	Volume	Integer	Volume
8	Bid	Decimal	Bid Price
9	Bid Size	Integer	Bid Size
10	Ask	Decimal	Ask
11	Ask Size	Integer	Ask Size
12	Last Trade Time	DateTime	Time last trade occurred YYYYMMDDHHMMSS
13	Bid Time	Time	Time bid quote received HHMMSS
14	Ask Time	Time	Time ask quote received HHMMSS
15	Quote Status	Integer	Quote status flags. (See Appendix A)
16	Market Condition	Numeric	Market Condition Code (See Appendix A)
17	Market ID	Decimal	Market Identifier (See Appendix A)
18	Event Time	Time	Time this event was sent. HHMMSS

Options Extension (only present for options records)

	Name	Type	Notes
19	Expiration Date	Date	Expiration
20	Strike Price	Decimal	Strike Price
21	Open Interest	Integer	Open Interest
22	Underlying	Alpha	Underlying asset symbol

Futures Extension (only present for futures records)

	Name	Type	Notes
19	Expiration Date	Date	Expiration
20	Open Interest	Integer	Open Interest
22	Settlement Price	Decimal	Settlement Price
22	Settlement Date	Date	Settlement Date

5.9 Option Chain Event

The Option Event occurs as a response to a GETOPTIONCHAIN or GETOPTIONCHAINL command on the socket interface. The fields of the Option Chain event contain the underlying as the event symbol followed by a list of the options for that underlying symbol.

	Name	Type	Notes
0	Message Type Symbol Type Symbol	"C"	Option Chain Event Type of symbol code. (See Appendix A) Security symbol (the underlying)
1	Option Symbol	String	First option symbol
	.		(option symbols)
	.		
	.		
	.		
N	Option Symbol	String	Last option symbol

5.10 Service Status Event

The Service Status Event is sent to notify applications of service status changes that are occurring or about to occur. Applications should monitor for this event to know when to reinitialize shared memory pointers and be ready for socket disconnections. The service restarts each day at the time specified in the configuration file.

	Name	Type	Notes
0	Message Type Symbol Type Symbol	"Z"	Service State Event (symbol type will be blank) (symbol will be blank)
1	Status Code	Integer	1 = Service will restart in 60 seconds 2 = Service is restarting now
2	Status Text	String	Text describing the status

6 Capture Files

TRAX: DTN RealTime can capture trades, quotes and minute bars for symbols of enabled databases. Which symbols are captured and whether trades, quotes and/or bars are captured is controlled by entries in the *dtn-realtime.conf* configuration file. (Section 2.3)

TRAX: DTN RealTime's capture facilities are designed to support charting and trading applications requiring minute bars and capturing data for back testing. However, it is not intended to provide full historical data capabilities for all applications. Use TRAX: DTN RealTime in conjunction with DTN IQFeed or another Internet market data service to get breadth of data available from the DTN satellite feed and full historical data.

Capture files are appended each day and may grow very large. It is the user's responsibility to truncate capture files to prevent the disk from becoming full.

6.1 System Resource Demands

Enabling capture of all market data can demand significant system resources. Be conservative in configuring capture. Only capture data that you will need and use. Capturing large amounts of data requires a fast processor, a fast disk subsystem and lots of disk space.

If you don't need or use quote history (bid/ask changes), don't enable capture for quotes. There are a large number of Bid/Ask changes that occur, especially for the Real Time Stock Option service.

One possible solution to aid with performance is to use a RAMDisk driver. The use of a RAMDisk is significantly faster than saving files to a hard disk drive.

6.2 Using Excel and Other Applications with Capture Files

Programs may use and open capture files without affecting TRAX as long as they don't open them in an exclusive mode.

Do not open capture files directly with Microsoft Excel or other applications that lock files for exclusive use. If a file is opened by such an application, TRAX: DTN RealTime will be unable to open the file to write new data. If you wish to open a capture file with such an application, first make a copy of the file, then open the copy - not the capture file.

6.3 Directory Structure

Each database-n section in the configuration file has a `CaptureDir` parameter. This is the sub-directory that TRAX will use to capture symbols for this database.

For example if the *dtn-realtime.conf* configuration file contains the lines:

```
[database-0]
CaptureDir = C:\DtnRealTime\capture\stock
```

The directories will be:

```
C:\DtnRealTime\capture\stock\bars
C:\DtnRealTime\capture\stock\ticks
```

6.3.1 Bars Capture

There are directories with the letters A-Z under the bars directory. These letters are the first alpha character of the symbols that are stored there. For instance, CSCO bar data will be stored in:

```
C:\DtnRealTime\capture\stock\bars\C\CSCO.csv
```

Symbols that begin with numerals will be stored in the directory comprised of the first alpha character, for instance 10MLIB (an index) would be stored in:

```
C:\DtnRealTime\capture\index\bars\M\10MLIB.csv
```

6.3.2 Tick Capture

For efficiency, ticks are captured in a single binary file, *ticks.bin*. Corrections are not written to the file (See Appendix D - Correction Indicator for more information on Corrections). The records in the tick file are in a binary format. Below is the C++ structure for tick records:

```
// Tick Types
#define TRAX_DTN_RT_TICK_TRADE 'T'
#define TRAX_DTN_RT_TICK_BID 'B'
#define TRAX_DTN_RT_TICK_ASK 'A'

struct TraxDtnRtTick
{
    char m_ucSymbolType;
    char m_pszSymbol[TRAX_DTN_RT_SYMBOL_LEN];
    BYTE m_ucTickType;
    BYTE m_ucDecimalCode;
    BYTE m_ucReserved;
    long m_lTime;
    int m_nPrice;
    int m_nSize;
};
```

Application programmers can read the file and process the tick records as required.

The *tick.bin* file must be maintained by the user and can grow very large. It is recommended that the user process the tick file each day to extract needed data, and then delete the file to free up disk space.

More support for processing tick data will be added to TRAX in the future.

6.4 File Structure

Bar files are stored in a simple format easily read by many programs. Each line is formatted as shown in the section below.

6.4.1 Bar Files

```
<datetime>,<open>,<high>,<low>,<close>,<volume>
```

The datetime field contains the date and time of the start of the bar in YYYYMMDDHHMM format where YYYY is the year, MM is the month, DD is the day of the month, HH is the hour and MM is the minute. The bar will contain data for seconds 0 through 59 for the minute designated. In the example below, the bar will contain data for 08:47:00 through 08:47:59

Example: 200401120847,27.68,27.6808,27.65,27.65,156600

7 Shared Memory Interface

The shared memory interface provides high performance access to real time market data for scanning the entire market or monitoring a large number of symbols. The real time records are organized in memory as an array or table. The header files and example code contain details of the structures, functions and procedures required to access the shared memory. Below is a portion of the C/C++ header file showing the structure of a single shared memory record. See the actual header file for the most current structure and for the extension structures used for options and futures.

```
//-----  
// TraxDtnRtMemoryRecord  
//-----  
struct TraxDtnRtMemoryRecord  
{  
    char m_pszSymbol[TRAX_DTN_RT_SYMBOL_LEN];  
  
    int m_nLastTradePrice;  
    int m_nLastTradeSize;  
  
    int m_nBidPrice;  
    int m_nBidSize;  
  
    int m_nAskPrice;  
    int m_nAskSize;  
  
    int m_nVolume;  
  
    int m_nOpen;  
    int m_nHigh;  
    int m_nLow;  
    int m_nPreviousClose;  
  
    int m_nLastTradePriceExt;           // Extended hours trade  
  
    long m_lLastTradeTime;              // Last trade reported  
    long m_lLastTradeRxTime;           // Last trade received  
    long m_lBidRxTime;                 // Last bid received  
    long m_lAskRxTime;                 // Last ask received  
  
    char m_cLastIndicator;  
    char m_cLastIndicatorExt;          // Extended hours trade indicator  
    char m_cBidTickIndicator;  
    char m_cUpcIndicator;              // 'N'=non-restricted, 'R'=restricted  
  
    BYTE m_ucDecimalCode;  
    BYTE m_ucMarketCondition;  
    BYTE m_ucMarketIdentifier;  
    BYTE m_ucMarketLastTrade;  
  
    BYTE m_ucMarketBestBid;  
    BYTE m_ucMarketBestAsk;  
    WORD m_wStatus;  
};
```

7.1 Shared Memory Fields

The table below gives descriptions of the shared memory fields. Applications should apply the data types as defined in section 5.1 to convert *Decimal* fields to actual floating point values.

Name	Bytes	Type	Notes
Symbol	12	String	Symbol of security
LastTradePrice	4	Decimal	Price of last trade
LastTradeSize	4	Integer	Size of last trade
BidPrice	4	Decimal	Bid price
BidSize	4	Integer	Bid size
AskPrice	4	Decimal	Ask price
AskSize	4	Integer	Ask size
Volume	4	Integer	Total volume traded today
Open	4	Decimal	Opening price
High	4	Decimal	High price today
Low	4	Decimal	Low price today
PreviousClose	4	Decimal	Previous close price
LastTradePriceExt	4	Decimal	Price of last extended hours trade
LastTradeTime	4	utime	Last trade reported time. Unix time format.
LastTradeRxTime	4	utime	Last trade received time. Unix time format. See Note A below.
BidRxTime	4	utime	Last Bid received time. Unix time format. See Note B below.
AskRxTime	4	utime	Last Ask received time. Unix time format. See Note B below.
LastIndicator	1	char	Last indicator. See Appendix A.
LastIndicatorExt	1	char	Last indicator for extended hours trade. See Appendix A.
BidTickIndicator	1	char	"U" - Up "D" - Down "N" - No Change " " - <space> Not indicated
UpclIndicator	1	char	"R" - Restricted " " - <space>
DecimalCode	1	BYTE	Decimal field code (See Appendix A)
MarketCondition	1	BYTE	Market Condition Code (See Appendix A)
MarketIdentifier	1	BYTE	Market Identifier (See Appendix A)
MarketLastTrade	1	BYTE	Market Identifier of last trade
MarketBestBid	1	BYTE	Market Identifier of best bid
MarketBestAsk	1	BYTE	Market Identifier of best ask
QuoteStatus	2	WORD	Quote status flags (See Appendix A)

Note A: The LastTradeTime is in seconds since January 1, 1970, coordinated universal time and must be adjusted to the Central Time zone. In ET zone, add 1 hour, MT subtract 1 hour, PT subtract 2 hours, etc.

Note B: : The BidRxTime and AskRxTime is in seconds since January 1, 1970, coordinated universal time.

Equity extension fields:

Name	Bytes	Type	Notes
PreviousVolume	4	Integer	Previous volume
AverageVolume	4	Integer	4 week average volume
MarketCap	4	Integer	Market capitalization (100,000's)
SharesOutstanding	4	Integer	Number of outstanding shares (1,000's)
PERatio	4	Decimal	Price/Earnings ratio
Beta	4	Decimal	Beta
Assets	4	Integer	Assets (1,000,000's)
Liabilities	4	Integer	Liabilities (1,000,000's)
Debt	4	Integer	Debt (1,000,000's)
AnnualEstEPS	4	Decimal	Annual estimated earnings per share
AnnualEPS	4	Decimal	Annual earnings per share
DilutedEPS	4	Decimal	Annual diluted earnings per share
EPS3YearGrowth	4	Decimal	Earnings per share 3 year growth
EPS5YearGrowth	4	Decimal	Earnings per share 5 year growth
PercentHeldInst	4	Decimal	Percent held by institutions
YtdHigh	4	Decimal	Year to Date High
YtdHighYear	2	WORD	YTD High Year (i.e., 2004, etc.)
YtdHighMonth	1	BYTE	YTD High Month (1 = Jan, 2 = Feb, etc)
YtdHighDay	1	BYTE	YTD High Day
YtdLow	4	Decimal	Year to Date Low
YtdLowYear	2	WORD	YTD Low Year (i.e., 2004, etc.)
YtdLowMonth	1	BYTE	Expiration Month (1 = Jan, 2 = Feb, etc)
YtdLowDay	1	BYTE	Expiration Day
52WeekHigh	4	Decimal	52 Week High
52WeekHighYear	2	WORD	52 Wk High Year (i.e., 2004, etc.)
52WeekHighMonth	1	BYTE	52 Wk High Month (1 = Jan, 2 = Feb, etc)
52WeekHighDay	1	BYTE	52 Wk High Day
52WeekLow	4	Decimal	52 Week Low
52WeekLowYear	2	WORD	52 Wk Low Year (i.e., 2004, etc.)
52WeekLowMonth	1	BYTE	52 Wk Low Month (1 = Jan, 2 = Feb, etc)
52WeekLowDay	1	BYTE	52 Wk Low Day
SICCode	2	WORD	Standard Industry Classification code
XdivIndicator	1	BYTE	Ex-dividend indicator ' ' - <space> no indication 'x' - See Note C below.
Reserved	1	BYTE	Reserved for alignment.
CurrentQuarterDiv	4	Decimal	Current quarterly dividend
AnnualDividend	4	Decimal	Annual dividend
DividendRate	4	Decimal	Dividend rate
DividendYield	4	Decimal	Dividend yield
ExDividendYear	2	WORD	Ex-Dividend Year (i.e., 2004, etc.)
ExDividendMonth	1	BYTE	Ex-Dividend Month
ExDividendDay	1	BYTE	Ex-Dividend Day
DividendPayYear	2	WORD	Date of Payment Year (i.e., 2004, etc.)
DividendPayMonth	1	BYTE	Date of Payment Month
DividendPayDay	1	BYTE	Date of Payment Pay Day
DividendRecYear	2	WORD	Date of Record Year (i.e., 2004, etc.)
DividendRecMonth	1	BYTE	Date of Record Month
DividendRecDay	1	BYTE	Date of Record Day
LastSplitFactor	4	Decimal	Last split factor (i.e., 0.5 = 2:1)
LastSplitYear	2	WORD	Last split Year (i.e., 2004, etc.)
LastSplitMonth	1	BYTE	Last split Month (1 = Jan, 2 = Feb, etc)
LastSplitDay	1	BYTE	Last split Day
PreviousSplitFactor	4	Decimal	Previous split factor (i.e., 0.5 = 2:1)
PreviousYear	2	WORD	Previous split Year (i.e., 2004, etc.)
PreviousMonth	1	BYTE	Previous split (1 = Jan, 2 = Feb, etc)

PreviousDay	1	BYTE	Previous split
YearEndClose	4	Decimal	Close price on Dec 31 of previous year
FiscalYEYear	2	WORD	Fiscal Year End Year (i.e., 2004, etc.)
FiscalYEMonth	1	BYTE	Fiscal YE Month (1 = Jan, 2 = Feb, etc)
FiscalYEDay	1	BYTE	Fiscal YE Day

Note C: The stock is trading without the right to receive the cash dividend. In this case, ExDividendYear,Month,Day will contain the next projected date that the security can be sold without the right to receive the cash dividend

Options extension fields:

Name	Bytes	Type	Notes
OpenInterest	4	Integer	Open Interest
Strike	4	Decimal	Strike price
ExpirationYear	2	WORD	Expiration Year (i.e., 2004, etc.)
ExpirationMonth	1	BYTE	Expiration Month (1 = Jan, 2 = Feb, etc)
ExpirationDay	1	BYTE	Expiration Day

Options extension extension fields:

Name	Bytes	Type	Notes
PreviousVolume	4	Integer	Previous volume

Futures extension fields:

Name	Bytes	Type	Notes
OpenInterest	4	Integer	Open Interest
ExpirationYear	2	WORD	Expiration Year (i.e., 2004, etc.)
ExpirationMonth	1	BYTE	Expiration Month (1 = Jan, 2 = Feb, etc)
ExpirationDay	1	BYTE	Expiration Day
SettlementPrice	4	Decimal	Settlement price
SettlementYear	2	WORD	Settlement Year (i.e., 2004, etc.)
SettlementMonth	1	BYTE	Settlement Month (1 = Jan, 2 = Feb, etc)
SettlementDay	1	BYTE	Settlement Day
VolumeIndicator	1	BYTE	Indicates volume type.
Reserved	3	BYTE	Reserved for alignment.

7.2 Accessing Shared Memory

The C++ code shown below details the steps necessary to gain access to a shared memory area.

```
char pszFileName[MAX_PATH];
sprintf(pszFileName, "C:\\market\\memory\\index");

// Open the memory backed file
DWORD dwShareMode = FILE_SHARE_READ | FILE_SHARE_WRITE | FILE_SHARE_DELETE;
HANDLE hFile = CreateFile(pszFileName, GENERIC_READ, dwShareMode, NULL,
OPEN_EXISTING, FILE_ATTRIBUTE_READONLY, NULL);
if (hFile == INVALID_HANDLE_VALUE)
{
    printf("Could not open shared memory file.\n");
    return 1;
}

// Open file-mapping object with read-only permission.
HANDLE hMapFile = CreateFileMapping(hFile, NULL, PAGE_READONLY, 0, 0, NULL);
if (hMapFile == NULL)
{
    printf("Could not create file-mapping object.\n");
    return 1;
}

// Map view of the file into our address space with read-only permission.
void* pMapAddress = MapViewOfFile(hMapFile, FILE_MAP_READ, 0, 0, 0);
if (pMapAddress == NULL)
{
    printf("Could not map view of file.\n");
    return 1;
}
```

7.3 Special Status Record

The first record in each shared memory section is a special service status record instead of data for a security. This aids the application developer in monitoring the status of the service through the shared memory. The C++ structure below shows the status record.

```
struct TraxDtnRtStatusMemoryRecord
{
    char m_pszSymbol[TRAX_DTN_RT_SYMBOL_LEN];    // "STATUS"

    long m_lStartTime;
    long m_lHeartbeatTime;
    long m_lReceivePackets;
    long m_lRecordType;
    long m_lRecordSize;
    long m_lMaxRecords;
};
```

The pseudo symbol will be set to "STATUS". The *StartTime* is the timestamp (UNIX timestamp format) of when the shared memory was initialized. An application should save this value in a program local variable and then periodically monitor the shared memory status record for a difference in it and the stored value to determine if the service has been restarted. If the service is restarted the application should reinitialize any cached pointers to shared memory records.

The *HeartbeatTime* is updated approximately once every 2 seconds and can aid an application in determining if the service has stopped for any reason.

The *ReceivePackets* field is updated with the receive packet count from the DTN receiver and can aid an application in determining that data is no longer being received from the DTN receiver. This could indicate a satellite outage or connection problem to the DTN receiver.

RecordType indicates the type of records contained in the shared memory section and will contain one of the following values:

```

const long TRAX_DTN_RT_SM_RECORD_TYPE_BASE = 1;
const long TRAX_DTN_RT_SM_RECORD_TYPE_OPTION = 2;
const long TRAX_DTN_RT_SM_RECORD_TYPE_FUTURE = 3;
const long TRAX_DTN_RT_SM_RECORD_TYPE_EQEXT = 4;
const long TRAX_DTN_RT_SM_RECORD_TYPE_OPTEXT = 5;

```

Also see the section 8.6 Shared Memory Descriptions.

RecordSize contains the size of a single record, while *MaxRecords* contains the maximum number of records available in the shared memory. TRAX allocates 100 additional records at the beginning of each day for any intraday symbol additions.

Scanning applications should skip over the Status Record to start scanning.

7.4 Scanning Market Data

Once a pointer to the desired shared memory area is obtained, check the first character of the *m_pszSymbol* field, it indicates the end of the table has been reached, otherwise points to a valid record and data may be directly accessed. Increment the pointer one record to advance to the next record. Below is a simple example in C++ of how to scan the table. Remember to skip the first record which is the special status record.

```

// Scan the shared memory table
TraxDtnRtMemoryRecord* pRecord = (TraxDtnRtMemoryRecord*)pMapAddress;
pRecord++; // skip status record
while (pRecord->m_pszSymbol[0] != '\0')
{
    double dMultiplier = dDecimalMultiplier[pRecord->m_ucDecimalCode];
    double dPrice = pRecord->m_nLastTradePrice * dMultiplier;
    printf("%10s %10.4f", pRecord->m_pszSymbol, dPrice);
    pRecord++;
}

```

All records are contiguous, but are not sorted. Applications may wish to scan the table and build their own lists, hashes, binary trees or data structures in order to support advanced search algorithms and other application specific needs.

In memory, the record table is an array of fixed width records.

[0]	Record 0
[1]	Record 1
[2]	Record 2
[3]	Record 3
	.
	.
	.
[N]	Record N

Each record in shared memory will remain at a fixed memory location until the TRAX: DTN RealTime service is terminated. The service reloads each night at the time scheduled in the *dtn-realtime.conf* configuration file. If applications run overnight, they should be aware that all pointers to the shared memory will become invalid at this time. The record for a given symbol - for instance CSCO - may be at location 0x0145950, but after the daily restart, the CSCO data may now be at memory location 0x0145320.

Applications may access the shared memory file even though the TRAX service may not be running. Be aware that if TRAX is not running, that the shared memory is *not* being actively updated - and your application may be reading stale (i.e., old) data.

8 System Files

TRAX creates and maintains a number of system files. These files provide operational data and market data for applications. These files are maintained in the directory specified in the configuration file (*dtn-realtime.conf*) for *FilesPath*.

8.1 Symbol Files

Symbol files are created each time TRAX initializes. For stocks, indexes and DTN calculated indicators, a file is written with the symbol and the issuer (company name) or description of the symbol. The symbol files are in CSV format:

```
<symbol>,<description>
```

The file names take on the DTN database names in lower case, with spaces replaced with hyphens. For instance, for NASDAQ STOCKS, the filename will be:

```
nasdaq-stocks.csv
```

For futures, the symbol file will not contain the description field:

```
<symbol>
```

For instance a line in the merc-futures.csv file might be:

```
@ESH4
```

8.2 Options Files

TRAX creates several files for options.

The file, ***options-strike-exp-all.csv***, contains a list of all options including LEAPS and includes the underlying symbol, option symbol, expiration date (YYYYMMDD) and strike price.

```
<underlying_symbol>,<option_symbol>,<exp_date>,<strike>
```

The file, ***options-strike-exp.csv*** is in the same format but contains a list of all options excluding LEAPS .

The file, ***option-roots.csv***, contains a list of all options and includes the underlying symbol, option root and option type.

```
<underlying_symbol>,<option_root>,<option_type>
```

Option type 0 is a short term option, type 8 is a LEAP.

8.3 Market Identifiers

The file, ***market-identifiers.csv*** contains a list of the Market Identifiers used in Socket Events, a symbolic name (Market ID) and a short description:

```
<market_code>,<market_id>,<description>
```

For example:

```
14,NYSE,New York Stock Exchange
```

8.4 Market Conditions

The file, **market-conditions.csv** contains a list of Market Conditions, the Market Condition Code (used in Socket Events), a symbolic name and a description:

<market_condition_code>,<symbolic_name>,<description>

Example:

4,MC_HALT,Trade Halt - Temporary halt in trading in a particular security for a participant.

8.5 Field Names

The file, **field-names.csv** contains a list of Field Names and their Field ID code as present in Field Update events. Please note that this is the complete list of DTN field identifiers. TRAX returns only the Field IDs listed in section 5.6.1 in Field Update events.

<field_id>,<field_name>

For example:

47,PE_RATIO

8.6 Shared Memory Descriptions

The file, **shared-memory.csv** contains a list of the user configured shared memory information. Some applications may require this to discover the user configured information and to obtain the shared memory record format. See the *TraxDtnRt.h* header file for specific record structures.

<DTN_database_name>,<memory_type>,<memory_file>

The memory types are:

Type	Description
base	Uses the base TRAX shared memory record type. <i>TraxDtnRtEMemoryRecord</i>
eqext	Uses the TRAX shared memory record type with the equity extension fields. <i>TraxDtnRtEquityExtMemoryRecord</i>
option	Uses the TRAX shared memory record type with the option extension fields. <i>TraxDtnRtOptionMemoryRecord</i>
optext	Uses the TRAX shared memory record type with the option extension extension fields. <i>TraxDtnRtOptionExtMemoryRecord</i>
future	Uses the TRAX shared memory record type with the future extension fields <i>TraxDtnRtFutureMemoryRecord</i>

For example:

NYSE STOCKS,base,C:\Program Files\Trax\memory\stock
REAL TIME STOCK OPTIONS,option,C:\Program Files\Trax\memory\option
MERC FUTURES,future,C:\Program Files\Trax\memory\future

8.7 Log Files

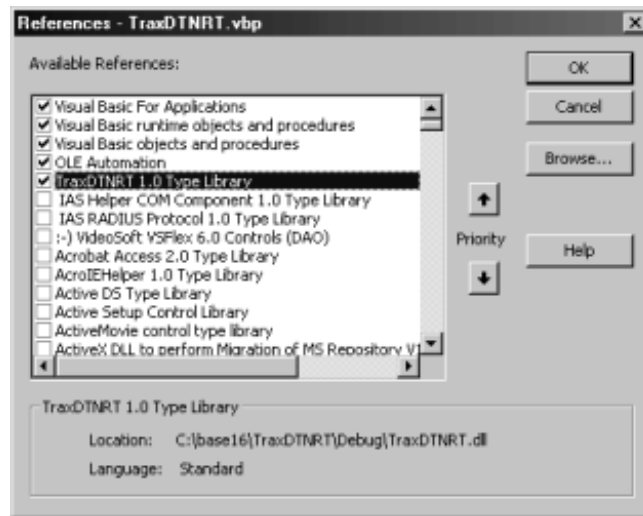
TRAX maintains a daily log file to record various service events and errors. The daily log file is maintained in the directory specified in the configuration file (*dtn-realtime.conf*) for *LogFilePath*.

The current day's log file is named ***dtn-realtime-log.txt***. Each day the log file is renamed to a date-coded filename. These log files should be periodically deleted by the user.

9 COM Socket Interface

A COM interface is provided with TRAX to allow Visual Basic and other applications to connect and send and receive data over the Socket Interface. To use this COM interface in your Visual Basic project:

- 1) From the Visual Basic menu, click on **Project->References**. Check the **TraxDTNRT 1.0 Type Library** from the Available References list then click OK.



- 2) Declare a variable name for the TraxDTNRT.Socket as a global variable:

```
Public WithEvents TraxSocket As TraxDTNRT.Socket
```

- 3) Instantiate the TRAX socket object:

```
' Create the TRAX socket object  
Set TraxSocket = New TraxDTNRT.Socket
```

- 4) Set the ApplicationName property and Connect

```
TraxSocket.AppName = "VB Example"  
TraxSocket.Connect
```

- 5) Handle the OnConnection and OnReceive events. You'll receive an OnConnection event of cstConnected signaling that a connection has been established with the TRAX Socket Interface. OnReceive events will contain the TRAX events described in the Socket Interface section of this manual. Your application must parse the event and take appropriate action.

- 6) Call Disconnect to close the connection.

See the Visual Basic socket example project included with TRAX for details on parsing and decoding event messages.

This interface is contained in the *TraxDTNRT.dll*.

9.1 Connect method

Connect to the TRAX Socket Interface.



You can use the Object Browser to explore the TraxDTNRT interface.

Be sure that the following properties are set before calling *Connect*. Address, Port, Application Name.

9.2 Application Name property

Set an application name to identify your application in the socket list. Limit is 50 characters. For more details, see the section 4.3.11 NAME in the Socket Interface.

9.3 Address property

The IP Address of the TRAX Socket Interface. The default address is 127.0.0.1 .

9.4 Port property

The TCP port of the TRAX Socket Interface. The default port is 8700 .

9.5 Connection State property

The connection state (status) of the TRAX Socket Interface. See *OnConnection* event for details.

9.6 Disconnect method

Disconnect from the TRAX Socket Interface.

9.7 SendCommand method

Send a command to the TRAX Socket Interface. Format the command as indicated in the Socket Interface section of this guide.

9.8 OnReceive event

A Socket event (data) was received from the TRAX Socket Interface. The event will be an Event Message as described in the Socket Interface section of this guide.

9.9 OnConnection event

The socket connection status has changed. Applications should monitor this event to be aware of connection status changes.

Connection Status	Description
cstDisconnected	The socket has disconnected.
cstConnected	The socket is connected.
cstClosed	The socket has closed.
cstFailed	The connection failed.

10 COM Shared Memory Interface

A COM interface is provided with TRAX to aid Visual Basic and other applications when using the Shared Memory Interface. Currently, this DLL provides a container organized as a binary tree to map symbols to shared memory record pointers. Shared Memory is directly addressed using record pointers. Please see the example Visual Basic projects provided with TRAX for more details.

To use this COM interface in your Visual Basic project:

1) From the Visual Basic menu, click on **Project->References**. Check the **TraxMemoryCOM 1.0 Type Library** from the Available References list then click OK.

2) Declare a variable name for the `TraxMemoryCOM.RecordManager` as a global variable:

```
Public TraxOptionRecordManager As TraxMemoryCOM.RecordManager
```

3) Instantiate the TRAX RecordManager object:

```
' Create the TRAX record manager object  
Set TraxOptionRecordManager = New TraxMemoryCOM.RecordManager
```

4) Scan the shared memory and locate any records that you wish to quickly access in the future and add them to the RecordManager using the `RecordManager.Add` method. Typically, you would add all symbols for a given shared memory area.

Use multiple instantiations of the RecordManager to manage more than one shared memory area.

5) Call the `RecordManager.Find` method to quickly lookup a pointer for a given symbol.

This interface is contained in the *TraxMemoryCOM.dll*.



You can use the Object Browser to explore the TraxMemoryCOM interface.

10.1 Add method

Add a symbol/pointer pair to the RecordManager map.

10.2 Find method

Return the pointer to the shared memory record of the given symbol.

11 RTD Server

The Real Time Data server included with TRAX: DTN RealTime allows users of Microsoft Excel 2002/ XP to easily access real time market data in their spreadsheet applications. (An example spreadsheet, **trax-example.xls** is included with TRAX.)

It's as easy as using a function in Excel.

- 1) From Excel's **Insert** menu choose **Function**.
- 2) Choose **RTD** in the Select a function list. The Function Arguments dialog as shown below will appear.
- 3) For **ProgID** enter "**RtdTrax.Server**"
- 4) Leave the Server field blank.
- 5) Enter the **symbol** you wish to monitor for **Topic1**. For instance in this example we're using CSCO. Use a period in front of a symbols to indicate an option. Also, some DTN symbols begin with an @ sign, for these symbol use a single quote in the cell before the @ or otherwise Excel will think that you wish to call a function.
- 6) For **Topic2**, enter a supported **field name**. For instance in this example we're using "Last" to fill the cell with the last trade price. The supported field names are listed in the section following.

As in any Excel function, you may reference cells for each parameter.

The screenshot shows the 'Function Arguments' dialog box for the RTD function. The dialog has a title bar with a question mark and a close button. Inside, the 'RTD' function is selected. The arguments are as follows:

Argument	Value	Default
ProgID	"RtdTrax.Server"	"RtdTrax.Server"
Server		text
Topic1	"CSCO"	"CSCO"
Topic2	"Last"	"Last"
Topic3		text

Below the arguments, there is a description: "Retrieves real-time data from a program that supports COM automation." and a note: "Topic3: topic1, topic2, ... are 1 to 28 parameters that specify a piece of data." At the bottom, the 'Formula result =' is shown as '26.20000076'. There are 'OK' and 'Cancel' buttons at the bottom right, and a 'Help on this function' link at the bottom left.

11.1 Security

A common initial problem you may encounter when using the RTD function in Excel is a problem with your Excel security setting. If the security level is set to High, then all you'll see is #N/A in cells that use the RTD function.

For TRAX you should set your security level to Medium or Low. To view or change your security level in Excel, on the Tools menu, point to Macro, click Security, and click the Security Level tab.

11.2 Refresh Interval

The TRAX RTD server uses a 200 millisecond internal timer to check for changed values in shared memory. If a value has changed, it will notify Excel that an update is needed.

Excel checks to see if it has been notified of an update using an internal value called `Application.RTD.ThrottleInterval`. This value is in Excel and can only be modified via the Excel object model or the registry. There is no user interface for configuring the RTD throttle interval in Excel.

If the RTD throttle interval is set to -1, this is considered manual mode, and Excel only checks for updates when `Excel.Application.RTD.RefreshData` is called. If the RTD throttle interval is set to zero, Excel checks for updates every chance it gets.

If the RTD throttle interval is set to something greater than zero, Excel waits at least that number of milliseconds between checks for updates.

Caution *If updates come in so frequently that Excel is continuously updating values and doing calculations, Excel might end up in a state where it never gives the user a chance to do anything, effectively getting in a hung state. If this happens, set the Excel throttle interval higher.*

To set the throttle interval higher through the Excel object model:

1. In Excel, go to the Visual Basic Editor (by pressing ALT+F11 or clicking **Visual Basic Editor** from the **Macro** menu (**Tools** menu)).
2. In the **Immediate** window (press CTRL+G or click **Immediate Window** on the **View** menu), type this code:

```
Application.RTD.ThrottleInterval = 1000
```

3. Make sure your cursor is on the line that you just typed, and then press ENTER.
4. To verify that it is set correctly, type this line of if code in the **Immediate** window:

```
? Application.RTD.ThrottleInterval
```

5. If you put your cursor at the end of this line and press ENTER, it should display 1000. Then you know that your throttle interval is set correctly.

To set the throttle interval higher through the registry, set the following registry key. It is a DWORD and is in milliseconds:

```
HKEY_CURRENT_USER\Software\Microsoft\Office\10.0\Excel\Options\RTDThrottleInterval
```

The above instructions are from the Microsoft web page **Real-Time Data: Frequently Asked Questions** at:

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnexcl2k2/html/odc_xlrtdfaq.asp

Please visit this web page for more details on using RTD.

11.3 RTD Fields

Below is a list of fields available for RTD:

Field	Description
OPEN	Open price
HIGH	High price for day
LOW	Low price for day
LAST	Last trade price
LASTTIMERX	Time last trade received
LASTEXT	Last trade price in extended hours trading
BID	Bid price
BIDSIZE	Bid Size (stocks only)
ASK	Ask
ASKSIZE	Ask Size (stocks only)
VOLUME	Volume
PREVVOL	Previous volume (stocks and options if memory extension only)
AVGVOL	4 week average volume (stocks if memory extension only)
PREVCLOSE	Previous Close
OPENINT	Open Interest (options and futures only)
STRIKE	Strike Price (options only)
EXPYEAR	Expiration Year (options and futures only)
EXPMON	Expiration Month (options and futures only)
EXPDAY	Expiration Day (options and futures only)
SETTLE	Settlement Price (futures only)
SETYEAR	Settlement Year (futures only)
SETMON	Settlement Month (futures only)
SETDAY	Settlement Day (futures only)

The special Status record fields (see section 7.3) can be monitored in Excel by using the following.

To monitor StartTime:

```
=RTD("RtdTrax.Server", , "STATUS", "LAST")
```

To monitor HeartbeatTime:

```
=RTD("RtdTrax.Server", , "STATUS", "LASTSIZE")
```

To monitor ReceivePackets:

```
=RTD("RtdTrax.Server", , "STATUS", "BID")
```

12 Wealth-Lab Adapter

TRAX includes a Real Time Data Adapter for Wealth-Lab Developer 3.0. Wealth-Lab is a complete platform for developing and back-testing stock and futures strategies based on technical analysis.

TRAX supports the **Charts System** for:

- Intraday Minute bars real time data and historical data.
- Tick real time data

TRAX supports the **Quotes System** for:

- Price, Size, Open, High, Low, Bid, Ask, Change, ChangePct.

Since TRAX does not manage End of Day (EOD) data, daily, weekly and monthly time frames are not supported.

To use the TRAX Real Time Data Adapter for Wealth-Lab:

1) Copy 4 files from your TRAX installation directory:

```
<install-dir>\Trax\bin\extras\Wealth-Lab
```

```
RTAdapter_TRAX.bmp  
RTAdapter_TRAX.conf  
RTAdapter_TRAX.dll  
RTAdapter_TRAX.txt
```

into your Wealth-Lab Developer 3.0 program directory and restart Wealth-Lab. The default Wealth-Lab program directory is:

```
C:\Program Files\Wealth-Lab\Wealth-Lab Developer 3.0
```

2) Select DataSources->Enable Live Feed->TRAX

3) Open a ChartScript window, and you should see TRAX in the DataSource tree. Click on any of the Intraday Intervals in the Time Scale toolbar. To get a tick interval, Select the pull down in the Scale Toolbar, and select Ticks. Tick history is not available from TRAX, but as trades are received, the chart will begin to fill with data.

If you have an options subscription, you'll also be able to chart intraday options data. Use a period before options symbols, for instance: .QQQBK

4) Select Tools->QuoteManager to access the QuoteManager in Wealth-Lab. Select Provider->TRAX. Use the QuoteManager as described in the Wealth-Lab Developer 3.0 User Guide.

12.1 Configuration File

The configuration file for the TRAX Real Time Data Adapter for Wealth-Lab is ***RTAdapter_TRAX.conf***. Edit this file with an ASCII editor. There is a single parameter in the file, `AutoFillStopTime`. This value limits the filling of minute bars with no trades at the end of the day. Set a time in 24 hour format to your local computer time.

13 Appendix A

Symbol Type codes and their descriptions are shown below.

Symbol Type Field	
Value	Description
[space]	Stock
[period]	Equity or Index Option
[dash]	Index or Calculated Values
[underscore]	Futures

Decimal codes and their descriptions are shown below.

Decimal Code Field	
Value	Description
0	Use value as-is.
1	Multiply by 0.1
2	Multiply by 0.01
3	Multiply by 0.001
4	Multiply by 0.0001
5	Multiply by 0.00001
6	Multiply by 0.000001
7	Multiply by 0.0000001
8	Multiply by 0.00000001

See section 8.3 for **Market ID** and section 8.4 for **Market Condition** values.

Status Field values for Event Messages and their descriptions are shown below. The indicators may appear in any order in the message.

Status Field	
Value	Description
*	Real Time - Market open
c	Real Time - Market closed
D	Delayed - Market open
d	Delayed - Market closed
R	UPC11830 Restricted
!	Halted
C	Correction (See Appendix D - Correction Indicator)
P	Post session
E	Recap
X	Expired
h	Daily high
l	Daily low (lowercase el)
H	52 week high
L	52 week low

Last Indicator Field values and their descriptions are shown below.

Last Indicator Field	
Value	Description
a	Ask
b	Bid
c	Market closed
.	Not trading yet (period)
U	Up
D	Down
-	No Change (dash)

Quote Status flags for the Shared Memory and their descriptions are shown below. These are bit-mapped fields.

Quote Status Flags	
Value	Description
Bit 4 - 0x0010	Daily Low Alarm - If set, this indicates that the current trade has set a new low for the day
Bit 5 - 0x0020	Daily High Alarm - If set, this indicates that the current trade has set a new high for the day
Bit 6 - 0x0040	52 Week Low/Contract Low - If set, this indicates that a new 52 week low has been reached.
Bit 7 - 0x0080	52 Week High/Contract High - If set, this indicates that a new 52 week high has been reached.
Bit 8 - 0x0100	Open - the instrument is open. Cleared when market closed.
Bit 10 - 0x0400	Instant - if set, data is real-time otherwise data is delayed
Bit 12 - 0x1000	Trading Halted - Trading has halted. Check Market Condition field for possible further reason or explanation.
Bit 13 - 0x2000	After hours trade
Bit 15 - 0x8000	Expired - For futures and options. Indicates the contract has expired

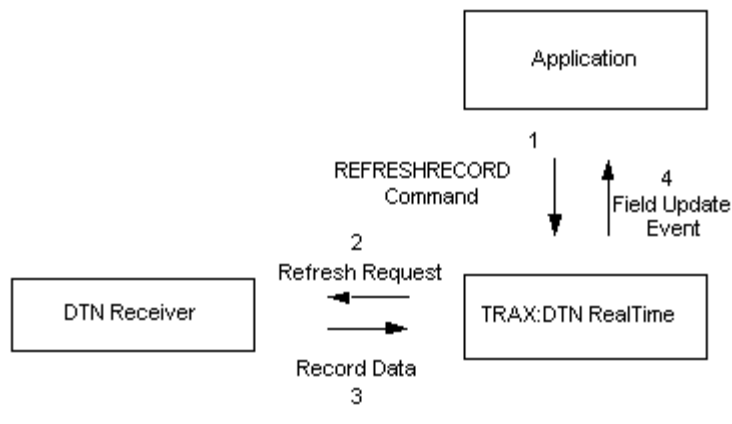
14 Appendix B - Receiver and Refreshing Record Data

Understanding how records are updated and received by TRAX: DTN RealTime is important to using the system.

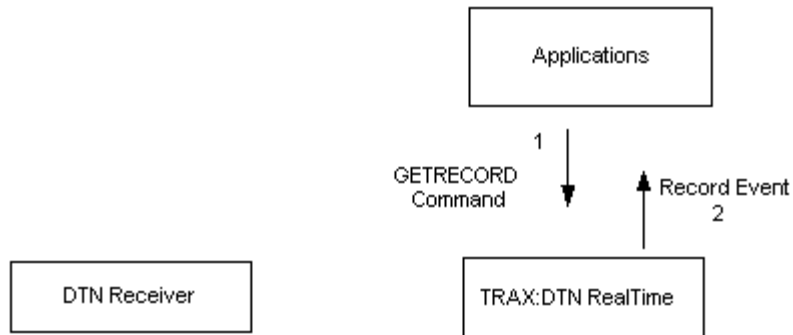
The diagram below shows the sequence of events that occur to retrieve the latest data from the receiver for REFRESHRECORD and REFRESHALLRECORDS commands. The same sequence occurs when Refresh or Refresh All Records are initiated from the TRAX: DTN RealTime Control program.

A similar command, GETRECORD, only retrieves data from the in-memory database, and is shown in the lower part of the diagram.

REFRESHRECORD Command



GETRECORD Command



15 Appendix C - DTN Supported Databases

INDEXES
INDEXES DELAYED

DTN CALCULATED INDICATORS
DTN CALCULATED INDICATORS DELAYED

NYSE STOCKS
AMEX STOCKS
NASDAQ STOCKS

REAL TIME STOCK OPTIONS

CBOT FUTURES
CBOT FUTURES DELAYED
MERC FUTURES
MERC FUTURES DELAYED
AIIT FUTURES
AIIT FUTURES DELAYED
KCBT FUTURES
KCBT FUTURES DELAYED
MGE FUTURES
MGE FUTURES DELAYED
WPG FUTURES
WPG FUTURES DELAYED
IPE FUTURES
IPE FUTURES DELAYED
CEC FUTURES
CEC FUTURES DELAYED
NYMEX FUTURES
NYMEX FUTURES DELAYED
COMEX FUTURES
COMEX FUTURES DELAYED
CBOT COMPOSITE FUTURES
CBOT COMPOSITE FUTURES DELAYED
COMEX COMPOSITE FUTURES
COMEX COMPOSITE FUTURES DELAYED
NYMEX COMPOSITE FUTURES
NYMEX COMPOSITE FUTURES DELAYED
MERC COMPOSITE FUTURES
MERC COMPOSITE FUTURES DELAYED
LONDON METALS EXCHANGE
LONDON METALS EXCHANGE DELAYED
LIFFE FUTURES
LIFFE FUTURES DELAYED
SIMEX FUTURES
SIMEX FUTURES DELAYED
EUREX FUTURES
EUREX FUTURES DELAYED
ENCOM FUTURES
ENCOM FUTURES DELAYED
ENIR FUTURES
ENIR FUTURES DELAYED
ENID FUTURES
ENID FUTURES DELAYED
HANNOVER FUTURES
HANNOVER FUTURES DELAYED

16 Appendix D - Correction Indicator

Trade Events and Field Update Events may contain corrections. The Status field will contain a Correction Indicator of "C" denoting that this is a correction and not a new trade. Applications should examine the Status field of each received Trade Event to determine if it is a correction or not.

Corrections are particularly important to the processing of Trade Events. If an application keeps a current "view" (cached values) of any of the fields in the Trade Event message, i.e., TradePrice, TradeVolume, TotalVolume, Trade Time, Trade Type, Last Indicator, Market ID, Status, the application should update this view using the information contained in a Trade Event with the Correction indicator. However, if the application is watching the stream for new trades (i.e., reading the tape), the application should not consider a Trade Event with the Correction Indicator as a new trade.

Corrections are always applied to the shared memory when received so that the data is as up to date and accurate as possible.

