Micro-line Canada Ltd.

Application Program Interface

Feed Reader Managed Service

Version 1.0

May 23, 2006

1. Notice

This specification is being provided strictly for information purposes, to assist in the development of systems to interact with Micro-line Canada Ltd. Feed Reader Managed Service.

2. Revision History

Version	Date	Author	Description
1.0	May 23, 2006	SO – Micro-line	Initial Version

TABLE OF CONTENTS

3. Preface

1.1 Scope

This guide describes the Application Program Interface to the Feed Reader Managed Service.

1.2 Audience

This guide is for developers building a programmatic interface to the Feed Reader Managed Service.

1.3 Typographical conventions

- Source code appears in `mono-spaced text` with a gray background. For example:

```
#/bin/sh
echo "started"
date
echo "finished"
```

- Important information or changes from previous versions of the document are highlighted.
- Directory names and filenames usually appear in a ***bold italic*** font. For example:

etc/populate_parameters_nexgen-primary.sql

- Commands typed by the user appear in `mono-spaced` font. For example:

  make upgrade

- Cross-references to other guides appear in regular italic enclosed in quotation marks. For example:

See the "*Administration – Feed Reader Managed Service*" document.

1.4 Definitions, Acronyms, and Abbreviations

API

The interface that a computer system, library or application provides in order to allow requests for service to be made of it by other computer programs, and/or to allow data to be exchanged between them (from http://wikipedia.org/).

BBK

Block Book trades data feed sourced by TSX Datalinx.

Berkeley Socket

The **Berkeley sockets** application programming interface (API) comprises a library for developing applications in the C programming language that perform inter-process communication, most commonly across a computer network.

Berkeley sockets (also known as the BSD socket API) originated with the 4.2BSD Unix operating system (released in 1983) as an API. Only in 1989, however, could UC Berkeley release versions of its operating system and networking library free from the licensing constraints of AT&T's copyright-protected Unix.

The Berkeley socket API forms the *de facto* standard abstraction for network sockets. Most other programing languages use a similar interface as the C API. (from http://wikipedia.org/).

CDL

CanDeal fixed income quotes data feed sourced by TSX Datalinx.

CL1

TSXV level 1 quotes and trades data feed sourced by TSX Datalinx.

CL2

TSXV level 2 orders and trades data feed sourced by TSX Datalinx.

CNQ

Canadian Trading and Quotation System trades, quotes, market by order and market by price data feed sourced by TSX Datalinx.

CNX

Canada NewsWire news releases headlines data feed sourced by TSX Datalinx.

COM

Component Object Model is a Microsoft platform for software componentry introduced by Microsoft in 1993. It is used to enable interprocess communication and dynamic object creation in any programming language that supports the technology. The term COM is often used in the software development world as an umbrella term that encompasses the OLE, OLE Automation, ActiveX, COM+ and DCOM technologies. Although COM was introduced in 1993, Microsoft did not begin emphasizing the name COM until 1997.

Although it has been implemented on several platforms, it is primarily used with Microsoft Windows. COM is expected to be replaced to at least some extent by the Microsoft .NET framework (from http://wikipedia.org).

DDE

Dynamic Data Exchange is a technology for communication between multiple applications under Microsoft Windows and also OS/2. Although still supported in even latest Windows versions, it has mostly been replaced by its much more powerful successors OLE, COM, and OLE Automation. However, it is still used in several places inside Windows, e.g. for Shell file associations (from http://wikipedia.org).

DLL

Dynamic-link library, also referred to as **dynamic link library** (without the hyphen), is Microsoft's implementation of the shared library concept in the Microsoft Windows operating systems. These libraries usually have the file extension DLL, OCX (for libraries containing ActiveX controls), or DRV (for legacy system drivers) (from http://wikipedia.org)..

DOS Shell

cmd.exe is the command line interpreter on OS/2 and on Windows NT-based systems (including Windows 2000, XP, and Server 2003). It is the equivalent of command.com in MS-DOS and Windows 9x systems, or of the shells used on Unix systems from (http://wikipedia.org).

Excel

Microsoft Excel is a spreadsheet program written and distributed by Microsoft for computers using the Microsoft Windows operating system and for Apple Macintosh computers. It features an intuitive interface and capable calculation and graphing tools which, along with aggressive marketing, have made Excel one of the most popular microcomputer applications to date. It is overwhelmingly the dominant spreadsheet application available for these platforms and has been so since version 5 in 1993 and its bundling as part of Microsoft Office (from http://wikipedia.org).

FAQ

Frequently Ask Questions - The term refers to listed questions and answers, all supposed to be frequently asked in some context, and pertaining to a particular topic (from http://wikipedia.org).

FX1

Foreign Exchange quotes data feed sourced by TSX Datalinx.

FX2

Foreign Exchange quotes, multiple contributors, data feed sourced by TSX Datalinx.

NE1

CP News stories, English language, data feed sourced by TSX Datalinx.

NF1

CP News stories, French language, data feed sourced by TSX Datalinx.

NW1

CP News stories data feed sourced by TSX Datalinx.

OS

An operating system is an essential software program that manages the hardware and software resources of a computer. The OS performs basic tasks, such as controlling and allocating memory, prioritizing the processing of instructions, controlling input and output devices, facilitating networking and managing files (from http://wikipedia.org).

RTD

Microsoft Excel 2002 provides a new worksheet function, RealTimeData, which allows you to call a COM Automation server for the purpose of retrieving data in real time.

When you need to create a workbook that includes data that is updated in real time -- for example, financial or scientific data -- you can now use the RTD worksheet function. In earlier versions of Excel, DDE is used for that purpose. However, the RTD function, which is based on COM technology, provides advantages in terms of robustness, reliability, and convenience. RTD depends on the availability of an RTD server to make the real-time data available to Excel.

The RTD function retrieves data from an RTD server for use in the workbook. The function result is updated whenever new data becomes available from the server and the workbook is able to accept it. The server waits until Excel is idle before updating, which relieves the developer of having to determine whether Excel is available to accept updates. The RTD function differs from other functions in this regard, because other functions are updated only when the worksheet is recalculated.

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

TCP

The Transmission Control Protocol (TCP) is one of the core protocols of the Internet_protocol_suite. Using TCP, applications on networked hosts can create

connections to one another, over which they can exchange data or packets. The protocol guarantees reliable and in-order delivery of sender to receiver data. TCP also distinguishes data for multiple, concurrent applications (e.g. Web server and e-mail server) running on the same host (from http://wikipedia.org).

TL1

TSX level 1 quotes and trades data feed sourced by TSX Datalinx.

TL2

TSX level 2 orders and trades data feed sourced by TSX Datalinx.

TSX

The Toronto Stock Exchange.

TSXV

The TSX Venture Exchange.

XL1

TSX indices data feed sourced by TSX Datalinx.

3.1 References

- *Toronto Level 1, Functional Specifications, TSX Markets, March 2005*
- Toronto Level 2 / TSX Venture Exchange Level 2, *Functional Specifications*, TSX Markets, September 2005
- TSX Venture Exchange Level 1, *Functional Specifications*, TSX Markets, March 2005
- Index Service Level 1, *Functional Specifications,* TSX Markets, October 2004
- CNQ MDF *Functional Specifications*, TSX Markets, February 2004
- CANDEAL *Functional Specifications*, TSX Markets, October 2004
- CNX Marketlink, *Functional Specifications*, TSX Markets, October 2003
- FX1, *Functional Specifications*, TSX Markets, December 2004
- MARKETS Inc. BBK Market Data Feed, *Functional Specifications*, April 2005
- FX2, *Functional Specifications*, TSX Markets, October 2005
- NW1, *Functional Specifications*, TSX Markets, October 2005

- TriAct Data Feed, Functional Specifications, Version 1.0, TSX Datalinx, February 2006.
- STAMP Specification, Version 4.0, TSX Markets, January 2003

1.5 Wikipedia License

Text copied from Wikipedia is made available under the following license:

http://en.wikipedia.org/wiki/Wikipedia:Text_of_the_GNU_Free_Documentation_License

4. Product Description

The Feed Reader Managed Service API provides:

- A TCP socket interface to one or more Toronto Stock Exchange (TSX/TSXV) level 1 (TL1, CL1, etc) feeds, level 2 (TL2, CL2) feeds, and other feeds such as BBK, CDL, CNQ, CNX, XL1, delivering a sequenced, reliable stream of data in field and record delimited format. This interface is operating system (OS) and compiler agnostic, and allows applications to be developed for any operating system and compiler that supports the establishment of TCP connections and sessions.
- A Microsoft Excel Real Time Data (RTD) interface to one or more Toronto Stock Exchange (TSX/TSXV) level 1 (TL1, CL1, etc) feeds. This interface is limited to the Microsoft OS and the Excel application. This interface is inherently unreliable due to the design of the Microsoft RTD interface as described below.

1.6 TCP Socket Interface

1.6.1 Overview

The Micro-line Feed Reader Service provides a TCP socket interface to each TSX feed. Client applications make a TCP connection to the Feed Reader Service IP address and per TSX feed port to receive a sequenced reliable stream of TSX feed data. The data is field and record delimited for easy manipulation by client applications.

1.1.1.1 Refer to the *"Administration – Feed Reader Managed Service"* document for the location of the Feed Reader Service, referred to as "**<MLFRS_01_IP>**" , "**<MLFRS_02_IP>", ..., "<MLFRS_NN_IP>**" below. The Feed Reader Service processing modules are assigned IP addresses on the client LAN, numbered from **01** to **NN**, where **NN** is the number of Feed Reader Service processing modules installed, usually **02**.

The following table shows the mapping of TSX feeds to ports. Note that while client applications can connect to any of these ports, data will be available only for those TSX feeds/ports which the client has purchased from TSX Datalinx. New TSX Datalinx service can be added at any time, this document will be updated to reflect this.

TSX Feed	***Business Content Format***	***Port***
BBK	FIX	9800

TSX Feed	*Business Content Format*	*Port*
CDL	Fixed length fields	9801
CL1	Fixed length fields	9802
CL2	STAMP	9803
CNQ	CNQ MDF	9804
CNX	Fixed length fields	9805
FX1	Fixed length fields	9806
FX2	Fixed length fields	9807
NE1	NewsML	9808
NF1	NewsML	9809
NW1	NewsML	9810
TL1	Fixed length fields	9811
TL2	STAMP	9812
XL1	Fixed length fields	9813

Table 1: TSX Feed Port Assignments

All TSX feed messages are framed with a transport header that identifies the feed and message type and provides a sequence number. The Feed Reader Service makes the transport header available to the application as 7 "**;**" delimited fields and then passes the Business Content through unchanged, except for Fixed length fields Business Content, which is passed through as 1 or more "**;**" delimited fields.

Refer to the TSX Datalinx per feed functional specification documents in the "**References**" section for details on processing the Business Content.

As well, each TSX feed has standard fixed length field circuit assurance messages that are passed through to the client application as "**;**" field delimited messages that are normally ignored by client applications.

Each feed message is newline (ASCII decimal 10, hex 0A) delimited.

The following table contains examples of messages received from the Feed Reader Service for some of the TSX feeds.

Feed	*Sample Message*
CL1	0067;000036361;CL1;0;0;A ;V ;XAU; ; ;00000600;000168;2;+;000038;2;033;007;121944 ;K
CL2	0226;000068601;CL2;0;0; ;V ;^A^^17=00000000^^54=80300304^^56=2006060212 191679^^50=59206^\^^6=OrderCancelResp^^5=Sel l^^70=36^^16=Cancelled^^40=S20060602000016^^ 196=0.000^^55=FO^^57=2006060212191663^^64=10 00^^247=CDX^^178=20060602114113859973^^165.0 =36
CDL	0129;000058364;CDL;0;0;F ;F ;CAN 3.750 06/01/2008 ;B;099345;3;004096;3;+;001000000;122238;0993 55;3;004091;3;- ;001000000;122403;002;002;B;F;B
TL1	0048;001097958;TL1;0;0;E ;T ;IPS; ; ;010571;2;004;010591;2;005;
TL2	0225;003252613;TL2;0;0; ;T ;^A^^17=00000000^^54=80300104^^56=2006060212 253129^^50=3236227^\^^6=OrderCancelResp^^5=B uy^^70=1^^16=Cancelled^^40=B20060602016289^^ 196=0.000^^55=GLG^^57=2006060212253116^^64=3 00^^247=TSE^^178=20060602122530151518^^165.0 =1

All Feed Reader Service messages start with the same 7 ';' delimited field header values:

Index	*Length*	*Description*
0	4	Message Length – length of the original message received from the TSX before procesing by the Feed Reader Service. Normally not used by a client application.

Index	*Length*	*Description*
1	9	Sequence Number - Every message is assigned a sequence number from 000000001 to 999999999 (decimal, ASCII), with wrap-around. The sequence number is incremented by 1 for each message sent. Normally not used by a client application. High availability applications would use this field if reading the same TSX feed from multiple Feed Reader Managed Service servers.
2	3	Service Identifier – Identifies the TSX feed per Table 1 above. Normally not used by a client application as the TCP port already identifies the TSX feed.
3	1	Re-transmission Identifier - Not used by a client application.
4	1	Continuation Identifier - Not used by a client application.
5	2	Message Type – Idenifies the type of message. Refer to the per feed TSX documentation for details.
6	2	Exchange Identifier – Identifies the Exchange. Refer to the per feed TSX documentation for details.

Using the TL1 message above as an example:

```
0048;001097958;TL1;0;0;E ;T ;IPS; ; ;010571;2;004;010591;2;005;
```

This consists of 17 **';'** delimited fields, the 7 header fields as described above and the remaining feed specific business content fields as described in the TSX TL1 Feed specification:

Field	*Description*
`0048`	Original length of message read from the TSX feed, not used.

Field	*Description*
001097958	9 digit sequence number, used if reading the same TSX feed from multiple Feed Read Managed Services servers for high availability applications.
TL1	This is a message from the TSX TL1 service.
0	Not used.
0	Not used.
'E '	This is a TSX TL1 '**Equity Quote**' message.
'T '	Exchange identifier, denoting the TSX exchange.
IPS	Stock Symbol Root.
' '	Type.
' '	Type descriptor.
010571	Bid Price.
2	Bid Price Fraction Indicator.
004	Bid Size in Boardlots.
010591	Ask Price.
2	Ask Price Fraction Indicator.
005	Ask Size in Boardlots.
' '	Markers.

Using the above TL2 message as an example:

```
0225;003252613;TL2;0;0;  ;T
;^A^^17=00000000^^54=80300104^^56=2006060212253129^^50=3236227^\^^6=OrderCancelResp^^
5=Buy^^70=1^^16=Cancelled^^40=B20060602016289^^196=0.000^^55=GLG^^57=2006060212253116
^^64=300^^247=TSE^^178=20060602122530151518^^165.0=1
```

This consists of the standard 7 '**;**' delimited header fields followed by the TL2 STAMP business content in the 8th field:

```
^A^^17=00000000^^54=80300104^^56=2006060212253129^^50=3236227^\^^6=OrderCancelResp^^5
=Buy^^70=1^^16=Cancelled^^40=B20060602016289^^196=0.000^^55=GLG^^57=2006060212253116^
^64=300^^247=TSE^^178=20060602122530151518^^165.0=1
```

Refer to the TSX TL2 feed specification for more detail.

For TSX feeds with Fixed Length Fields Business Format content, the Feed Reader Managed Service converts fixed length business content to '**;**' delimited fields. Note that the format is still fixed length on a per message basis so the client application can process the feed as fixed length or as field and record delimited.

For TSX feeds with any other type of Business Format content (CL2 – STAMP, CNQ – CNQ MDF, etc.), the Feed Reader Managed Service converts passes through the Business Format content unchanged.

For all TSX feeds, the Feed Reader Managed Service converts the TSX feed header to 7 '**;**' delimited fields and appends a new line (ASCII 10) to each message.

All TSX feeds send 2 types of heartbeat messages, one from the TSX retransmission service and one from the data transmission service. These messages are received once per minute from the TSX and are usually ignored by the application as the Feed Reader Managed Service manages the processing of these messages.

4.1 Connecting to the Feed Reader Service

The client application makes a TCP socket connection to one of the Feed Reader service IP addresses: "**<MLFRS_01_IP>**" , "**<MLFRS_02_IP>", ..., "<MLFRS_NN_IP>**"

The IP address to use is a client application design issue. For reliable service, the client application may be run in a redundant/hot standby mode such that multiple copies of the client application are run, each connecting to a different Feed Reader service IP. Alternatively, a single instance of the client application may connect to multiple Feed Reader service IP addresses and filter out duplicate messages using the **Sequence Number** field. Or different client applications may connect to different Feed Reader service IP addresses to balance the load on the Feed Reader Managed Service.

The client application makes a TCP connection using one of the ports described above in Table 1, per the TSX feed of interest.

Once the TCP connection is made, the client application will start receiving a stream of newline delimited messages from the Feed Reeder Managed Service.

4.2 Reading Messages from the Feed Reader Service

The client application reads the stream of feed data from the TCP feed socket per the socket API being used, i.e. **read**() or **recv**() if the Berkeley Socket API is being used.

The client should always be ready to read data from the socket as it arrives, either using the **select**() function to multiplex IO on multiple file descriptors or using a dedicated thread in a multi-threaded program.

Should the client application fail to read data from the feed socket in a timely manner, the Feed Reader Managed Service may disconnected the socket on its end if the the socket connection is consuming too many resources.

Typically the client application will not receive one complete feed message per call to **read**() or **recv**(). The client application must build complete newline delimited feed messages from the feed data stream.

Once a complete feed message is received, it can be tokenized into multiple '**;**' delimited fields and process as per the TSX feed specification.

4.3 Disconnecting from the Feed Reader Service

The application closes the TCP socket connection to the Feed Reader Service.

4.4 Example Program

A fully functional example Perl program that connects to the TL1 feed, reads TL1 messages until and **Equity Quote** message is read, displays the message, disconnects and terminates is presented below.

```
#!/usr/bin/perl -w

# feed_reader_api_example.pl - example program connects to TL1 feed, reads and
display the next 'Equity Quote' message, and disconnects

use strict;

no strict "refs";

use Socket;

(scalar(@ARGV) == 2) || die "usage: $0 <MLFRS_TL1_IP> <MLFRS_TL1_PORT>\n";


my $mlfrs_tl1_ip = $ARGV[0];

my $mlfrs_tl1_service = $ARGV[1];


my $port;

if($mlfrs_tl1_service =~ m/[0-9]+/){

    $port = $mlfrs_tl1_service;

}else{

    $port = getservbyname($mlfrs_tl1_service, 'tcp') || die "getservbyname() failed:
'$mlfrs_tl1_service' - $!\n";
```

```
} # if
my $iaddr = inet_aton($mlfrs_tl1_ip) || die "no host: '$mlfrs_tl1_ip' - $!\n";
my $paddr = sockaddr_in($port, $iaddr);
my $proto = getprotobyname('tcp');
socket(S, PF_INET, SOCK_STREAM, $proto)  || die "socket() failed - $!\n";
if(!connect(S, $paddr)){
    die "connect() failed: host '$mlfrs_tl1_ip' service '$mlfrs_tl1_service' - $!\n";
} # if
my $save_fh = select S; $| = 1; select $save_fh; # unbuffered I/O


my $line;
while($line = <S>){
    $line =~ s/[\n]//;
    my @f = split(/;/, $line);
    if(scalar(@f) == 17 && $f[5] eq 'E '){
        printf("TL1 Equity Quote: Symbol '%s' Type '%s' Type Descriptor '%s' Bid
Price '%s' Bid Price FI '%s' Bid Size '%s' Ask Price '%s' Ask Price FI '%s' Ask Size
'%s' Markers '%s'\n", $f[7], $f[8], $f[9], $f[10], $f[11], $f[12], $f[13], $f[14],
$f[15], $f[16]);
        last;
    } # if ... TSX TL1 Equity Quote message
} # while
close(S);
exit(0);
```

When this program is run from a shell as follows:

```
./feed_reader_api_example.pl mlfrs_ip 9811
```

the output is similar to:

```
TL1 Equity Quote: Symbol 'RIM' Type ' ' Type Descriptor ' '
Bid Price '007469' Bid Price FI '2' Bid Size '014' Ask Price
'007478' Ask Price FI '2' Ask Size '009' Markers ' '
```

1.7 Microsoft Excel RTD Interface

1.7.1 Overview

The Micro-line Feed Reader Service provides an Excel RTD interface to quote and trade data from the TSX CL1 and TL1 feeds.

The RTD interface consists of a DLL file and registry (.reg) definition files that are copied via HTTP from the Micro-line Feed Reader Service and installed on each client workstation that uses the RTD interface.

The Micro-line RTD interface is started by Excel each time a spreadsheet containing Micro-line RTD formulas is loaded.

1.7.2 Limitations

The Microsoft RTD Interface uses a push/pull model. The Micro-line RTD server informs Excel when there is an update available for a cell. Excel will ask the RTD server for updates when it is ready.

From the Microsoft RTD FAQ:

How Do I Make Sure That Excel sees Every Update?

It's basically up to the RTD server to decide what to do with the values if more than one update on a particular topic comes in before Excel calls back to the server for updates. Generally, we expect servers to just throw out the old value and pass the new value into Excel when it asks for updates. But there are some instances we've heard of where someone wants every update. In that case, the RTD server needs to queue up updates. When Excel asks for updates, the RTD server sends Excel the oldest one in the queue. The server continues calling the ***UpdateNotify*** *method while the queues still have values in them.*

The Micro-line RTD server throws away old values per Microsoft best practices for RTD. If it were to queue values for rapidly updating instruments, then the RTD server would present the client application with out of date data.

Any client application that needs guaranteed delivery of tick by tick market data should not be implemented in Excel but should use the **TCP Socket Interface** described above.

1.7.3 Hardware and Software Requirements

Microsoft Excel version 2002 (office XP) or above running under Microsoft Windows 98, ME, XP or 2000. The RTD service cannot be used on earlier versions of Excel as the real-time data technology required for a robust streaming service was first released in Excel 2002.

What Kind of Performance Statistics Have Been Seen With the New RTD Architecture?

On a Pentium III 500 MHz processor with 128 MB of RAM, 20,000 unique topics can be updated three times per second; one topic can be updated 200 times per second.

How Frequently Can Excel Accept Updates?

These are the numbers that we get with an RTD server running on the same computer that Excel is on.

Based on a Pentium III 500 MHz processor with 128 MB of RAM, 20,000 unique topics can be updated three times per second; one topic can be updated 200 times per second.

The number of times that a single topic can be updated seems to be limited by the number of times that Excel checks for Microsoft Windows messages, which is at most 700 times per second. Since some of the messages have higher priority than RTD does, Excel effectively gets about 200 updates per second.

1.7.4 Installation and Configuration

1.7.4.1 Location of Feed Reader Service

1.1.1.2 Refer to the *"Administration – Feed Reader Managed Service"* document for the location of the Feed Reader Service, referred to as "**<MLFRS_IP>**" below.

1.7.4.2 Create Local Directory/Upgrade

Create a directory on the client workstation called "**C:\MLFRS**".

If this directory already exists then this an upgrade. In this case, execute the following commands from a DOS shell to unload the existing RTD server after shutting down all Excel spreadsheets:

```
cd c:\MLFRS
regsvr32 /s /u MLRTDTL1Server.dll
```

1.7.4.3 Registry

The following registry files are copied from the Micro-line Feed Reader Service:

- http://<MLFRS_IP>/rtd.tl1.reg – CL1 and TL1 RTD server registry settings. Copy to the "**C:\MLFRS**" local directory on the client workstation and double click from Windows Explorer. This file contains IP addresses and ports that allow the RTD server to connect to the Feed Reader Service.
- http://<MLFRS_IP>/excel.rtdthrottleinterval.reg – Sets the Excel RTD throttle interval to zero milli-seconds so that TSX CL1 and TL1 quote and trade tics are delivered to Excel with no delay. The default value is 2000 mill-seconds, which is of no use to an application that needs tick by tick data in real time. Copy to the "**C:\MLFRS**" local directory on the client workstation and double click from Windows Explorer.

1.7.4.4 DLL

Copy the file "**http://<MLFRS_IP>/MLRTDTL1Server.dll**" the "**C:\MLFRS**" local directory. Execute the following commands from a DOS shell:

```
cd c:\MLFRS
regsvr32 /s MLRTDTL1Server.dll
```

These commands must be executed each time the client workstation is rebooted.

1.7.4.5 Set Excel Security Level

The behaviour of the RTD server is governed by Excel security settings. To examine or modify your security settings in Excel, on the **Tools** menu, click **Macro**, and select **Security**.

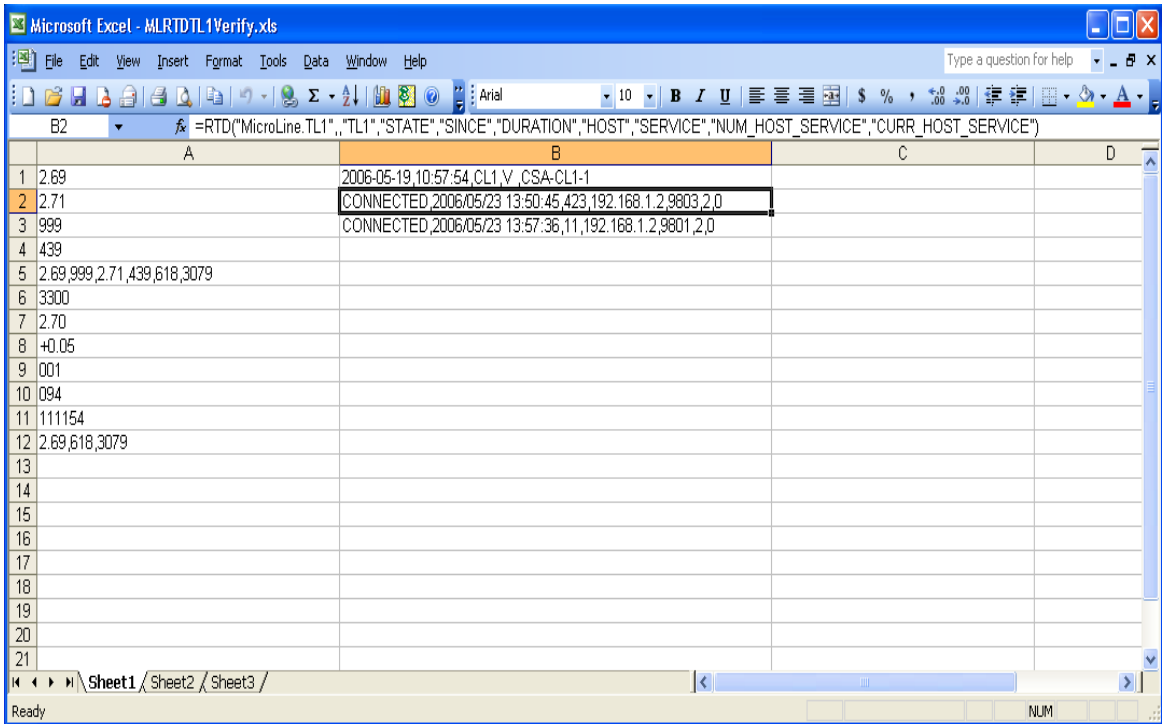
- When the security is set to **High**, the RTD server fails silently, and Excel displays #N/A in the worksheet cell.
- When the security is set to **Medium**, Excel 2002 displays a warning that the RTD server contains macros and asks the user whether to enable or disable them. If the macros are enabled, the RTD server runs normally. If the macros are disabled, the RTD server fails silently and Excel displays #N/A in the worksheet cell.
- When the security is set to **Low**, the RTD server runs normally without any security warnings.

Excel 2002 is designed to work in this manner for security purposes. Because an RTD server can be installed on a computer without the user's knowledge, and because an RTD server requires no user interaction to function, Excel 2002 notifies users when an RTD server is present and allows them to disable it if they wish.

1.7.4.6 Verify Installation and Troubleshooting

Copy the file "**http://<MLFRS_IP>/MLRTDTL1Verify.xls**" to the "**C:\MLFRS**" directory.

Load the file into Excel and compare to the diagram below:



Excel has connected to the Micro-line RTD server and is receiving updates. If this is being run during market hours (Mon – Fri 09:30 – 16:00), expect the see the cells update frequently.

If all the cells contain the value "**#N/A**" then there is a problem. Verify the installation steps above.

The most likely causes of the problem are:

- The registry values have not been set. Use **"regedit32"** to verify that the following values are set:

  1. HKEY_CURRENT_USER\Software\Micro-line\RTD\TL1\feed_reader_host – one or more comma separated IP addresses for the Feed Reader TL1 service.
  2. HKEY_CURRENT_USER\Software\Micro-line\RTD\TL1\feed_reader_service – one or more comma separated ports for the Feed Reader Tl1 service.
  3. HKEY_CURRENT_USER\Software\Micro-line\RTD\CL1\feed_reader_host – one or more comma separated IP addresses for the Feed Reader CL1 service.
  4. HKEY_CURRENT_USER\Software\Micro-line\RTD\CL1\feed_reader_service – one or more comma separated ports for the Feed Reader Tl1 service.

Re-run the registry files per the **Registry** section above.

Refer to the "*Administration – Feed Reader Managed Service*" document for details on valid IP addresses and ports for the above registry entries.

- Verify that Excel security setting is "**Medium**" or "**Low**".
- The file **"MLRTDTL1Server.dll**" has not been loaded. Run the command "**regsvr32 MLRTDTL1Server.dll**". A dialog box with the message "**DllRegisterServer in MLRTDTL1Server.dll succeeded**". Any other message indicates that the DLL is not loaded. Verify installation of the DLL as described above.
- Check the RTD server log file. This file is created in "**C:\MLRS**" each time the RTD server is started and has a name of the form "**tsx_rtd_server-YYYYMMDD-HHMMSS.log**", i.e. "**tsx_rtd_server-20060523-131547.log**". This file will only exist if the file "**MLRTDTL1Server.dll**" has been loaded. Look for error messages of the form:

```
2006/05/23 13:15:47;error;VCRTDServer()::ServerStart():
RegQueryValue(HKEY_CURRENT_USER, 'Software\Micro-line\RTD\TL1\feed_reader_host')
failed
```

In the above example, the RTD server is unable to locate a registry key.

1.7.5 Accessing CL1 and TL1 Data Using Excel

The general form of an Excel formula that provides access to the CL1 and TL1 data feeds is:

```
=RTD("MicroLine.TL1",,<TICKER>,<FIELD>,...)
```

For example:

```
=RTD("MicroLine.TL1",,"NT","BID","BIDSIZE")
```

displays the most recent "**BID**" and "**BIDSIZE**" field values for the ticker symbol "**NT**".

The "**http://<MLFRS_IP>/MLRTDTL1Verify.xls**" Excel verification spreadsheet demonstrates most of the RTD syntax described below.

The RTD function takes the following arguments:

1. A string that represents the Program ID of the RTD server installed on the local system. It is always "**MicroLine.TL1**" for both CL1(TSXV) and TL1(TSX) ticker symbols.
2. The second argument is always empty as the RTD server is running on the client workstation so the name of a remote computer is not used.
3. A valid CL1(TSXV) or TL1(TSX) symbol in TSX ticker format or predefined values that allow the status of the RTD server to be displayed as described below.
4. One or more fields associated with the ticker symbol as described below.

The RTD function returns the specified fields into a single cell with multiple comma separated values, one per field in the RTD function call. It is the responsibility of the Excel spreadsheet to parse this into multiple fields if needed or to request only one field per cell.

Ticker fields are associated with a quote or a trade.

The quote related fields are:

- BID – The current bid price.
- BIDSIZE – The number of boardlots represented by the bid price, 999 indicates 999 boardlots or more.
- ASK – The current ask price.
- ASKSIZE – The number of boardlots represented by the ask price, 999 indicates 999 boardlots or more.
- MARKERS_QUOTE – markers associated with the quote as follows:
  - 'A' - halted
  - <blank> - no markers
- UPDATE_QUOTE – since there is no TSX generated timestamp associated with a quote message, the Feed Reader service generates a timestamp in the form "**YYYY/MM/DD HH:MM:SS.mmmmmm**", i.e. "**2006/05/23 15:43:12.779216**", with micro-second resolution.
- COUNT – The total number of updates received for the ticker symbol from the Feed Reader service since the RTD server has been started by Excel.
- MISSED – The total number of updates for the ticker symbol received by the Feed Reader service since the RTD server has been started by Excel but missed by Excel, due to a new update being received from the Feed Reader service before Excel has requested the previous update.

The trade related fields are:

- TRDVOL_1 - Volume of shares traded.
- TRDPRC_1 - Price at which the transaction took place.
- NETCHG_1 - Net change = last trade price – previous close.
- BUYER_ID - Identifies the buying broker.
- SELLER_ID - Identifies the selling broker.

- STAMP_TIME – TSX generated time of trade in the format "**HHMMSS**".
- MARKERS_TRADE – markers associated with the trade as follows:
  - B – delayed trade
  - C – contingent trade
  - D – cash
  - E – non boardlot
  - F – mandatory cash
  - G – VWAP trade
  - K – set last price
  - L – set open price
  - M – special terms trading
  - O – basis trade
  - Q – MOC trade
  - S – special trading session
  - U – trading in $US
  - X – internal cross
  - <blank> - no markers
- UPDATE_TRADE – The Feed Reader service generates a timestamp in the form "**YYYY/MM/DD HH:MM:SS.mmmmmm**", i.e. "**2006/05/23 15:43:12.779216**", with micro-second resolution.
- COUNT – The total number of updates received for the ticker symbol from the Feed Reader service since the RTD server has been started by Excel.
- MISSED – The total number of updates for the ticker symbol received by the Feed Reader service since the RTD server has been started by Excel but missed by Excel, due to a new update being received from the Feed Reader service before Excel has requested the previous update.

The following special ticker symbols allow the state of the RTD server to be displayed. These are typically used for diagnostic purposes:

- HEARTBEAT – each CL1 and TL1 heartbeat generates an update for this special ticker symbol. The following fields are associated with this ticker symbol:

- DATE - Date in format YYYY-MM-DD.
- TIME - Time of day in format HH:MM:SS.
- FEED - "**CL1**" for CL1(TSXV) heartbeat, "**TL1**" for TL1(TSX) heartbeat.
- EXCHANGE - **"V** " for CL1(TSXV) heartbeat, "**T** " for TL1(TSX) heartbeat.
- TSX_SOURCE – identifies the TSX site that is sourcing the data:
  - CSA-CL1-1 - CL1 data sourced from the TSX Markham site.
  - CSA-CL1-2 - CL1 data sourced from the TSX Toronto site.
  - CSA-TL1-1 - TL1 data sourced from the TSX Markham site.
  - CSA-TL1-2 - TL1 data sourced from the TSX Toronto site.

- CL1 or TL1 – the state of the "**CL1**" or "**TL1**" feed. The following fields are associated with these ticker symbols:
  - STATE – One of "**CONNECTED**" or "**DISCONNECTED**", indicating whether the RTD server is currently connected to the "**CL1**" or "**TL1**" feed.
  - SINCE – The date/time that the RTD server entered the above STATE for the "**CL1**" or "**TL1**" feed. The format is "**YYYY/MM/DD HH:MM:SS".**
  - HOST – The hostname/IP address that the RTD server is currently connected to for the "**CL1**" or "**TL1**" feed if STATE is "**CONNECTED**" or the hostname/IP address that the RTD server last tried to connect to if STATE is "**DISCONNECTED**".
  - SERVICE - The service/port that the RTD server is currently connected to for the "**CL1**" or "**TL1**" feed if STATE is "**CONNECTED**" or the service/port that the RTD server last tried to connect to if STATE is "**DISCONNECTED**".
  - NUM_HOST_SERVICE – The number of host/service pairs that the RTD server is configured to make connection attempts to as per the RTD server registry settings described above.
  - CURR_HOST_SERVICE – The index of the host/service pairs that the RTD server is currently connected to for the "**CL1**" or "**TL1**"

feed if STATE is "**CONNECTED**" or the index of the host/service pairs that the RTD server last tried to connect to if STATE is "**DISCONNECTED**". The values range from 0 (zero) to (NUM_HOST_SERVICE – 1).

- DURATION – The time in seconds since the RTD server entered the above STATE for the "**CL1**" or "**TL1**" feed.