ICE iMpact Multicast Feed
Getting Started

Version 1.1.17

Table of Contents

Version	Date	Description of Changes
1.0.01	Feb 2008	Initial draft
1.0.02	Feb 2008	Bug Fixes
1.0.03	Feb 2008	Fixed bugs causing the price levels to be stuck. Consolidated the ready queue. Minor logging changes. Menu Option to get the Open Interest
1.0.04	Feb 28, 2008	Included Multicast Monitor and the log4j configuration. Overall packaging directory structures changed. Remove client_example dir. New lib/conf directories added. Additional jar files. Common.bat file included.
1.0.05	Mar 4, 2008	Included a configurable property, used to specify the network interface to listen on, for the multicast feed. Renamed the system properties for consistency
1.0.06	July 24, 2008	Include the section on OS-specific information
1.0.07	July 24, 2008	Use of the XML based Multicast Client configuration file
1.0.08	Nov 24, 2008	Change TCP server port from 2000 to 3000 for Test environment
1.0.09	Dec 12, 2008	Update version number to match the latest spec.
1.0.09	Jun 18, 2009	Update non-production environment connection information
1.0.09	June 29, 2010	Update IP address of the TCP connection to APITest
1.1.12	Apr 27, 2012	Update version number to match main message spec
1.1.16	Mar 19, 2014	Update version number to match main message spec
1.1.17	Mar 20, 2014	Update version number to match main message spec

1. Document Overview

What is the ICE iMpact Multicast Feed?

The ICE iMpact **Multicast** Feed delivers real time market data messages in a platform independent format through multicast. Please read "ICE iMpact **Multicast** Feed Technical Specification" for more details.

What is included in the "Getting Started" package?

```
ICE_iMpact_ Multicast _Feed_FAQ.pdf
ICE_iMpact_ Multicast _Feed-Getting_Started.pdf
ICE_iMpact_ Multicast _Feed_Spec.pdf
ICE_iMpact_ Multicast _Feed_Message_Spec.pdf
ICE_iMpact_ Multicast _Feed-Wireshark_Dissector.pdf


src/
        Example Java Source Code for the Simple Multicast Client, Multicast GUI Client
bin/
        common_bat.txt
        common.sh

        lib/
                pricefeedclient.jar
                log4j.jar
                mail.jar
                activation.jar
                SNMP4J.jar
                ICELogger.jar

        client/
                multicastClient_bat.txt (rename it to .bat)
                multicastClient.sh

                conf/
                        multicastClientConfig.xml
                        log4j.xml (Java logging)

        simple/
                simpleMulticastClient_bat.txt (rename it to .bat)
                simpleMulticastClient.sh
                simpleTcpClient_bat.txt (rename it to .bat)
                simpleTcpClient.sh

                conf/
                        simpleTcpClient.properties
        tunnel/
                tunnelProxy_bat.txt (rename it to .bat)
                tunnelProxy.sh

                conf/
                        tunnelProxy.properties
                        log4j.xml (Java logging)

        monitor/
                multicastMonitorClient_bat.txt (rename it to .bat)

                conf/
                        log4j.xml (Java logging)
Wireshark/
        icemulticast.dll
```

[Please note that the Java source code examples included as part of this package are simply for reference only and will not be supported for production purposes. Also note that while these examples are being provided in Java, the iMpact data feed is platform independent and your client may be developed in your language of choice.]

What are the requirements of running the sample code?

JDK 1.6

2. Test Environments

The iMpact multicast feed is currently available for testing on API Test environment. First, you would need to have connectivity to the TCP server to request for product definitions or historical replay. The followings are the IP address and port for the TCP server, which can be accessed through the Internet. You might need to have your firewall configured to access the server.

TCP Server IP Address:	63.247.113.163
TCP Server Port:	3000

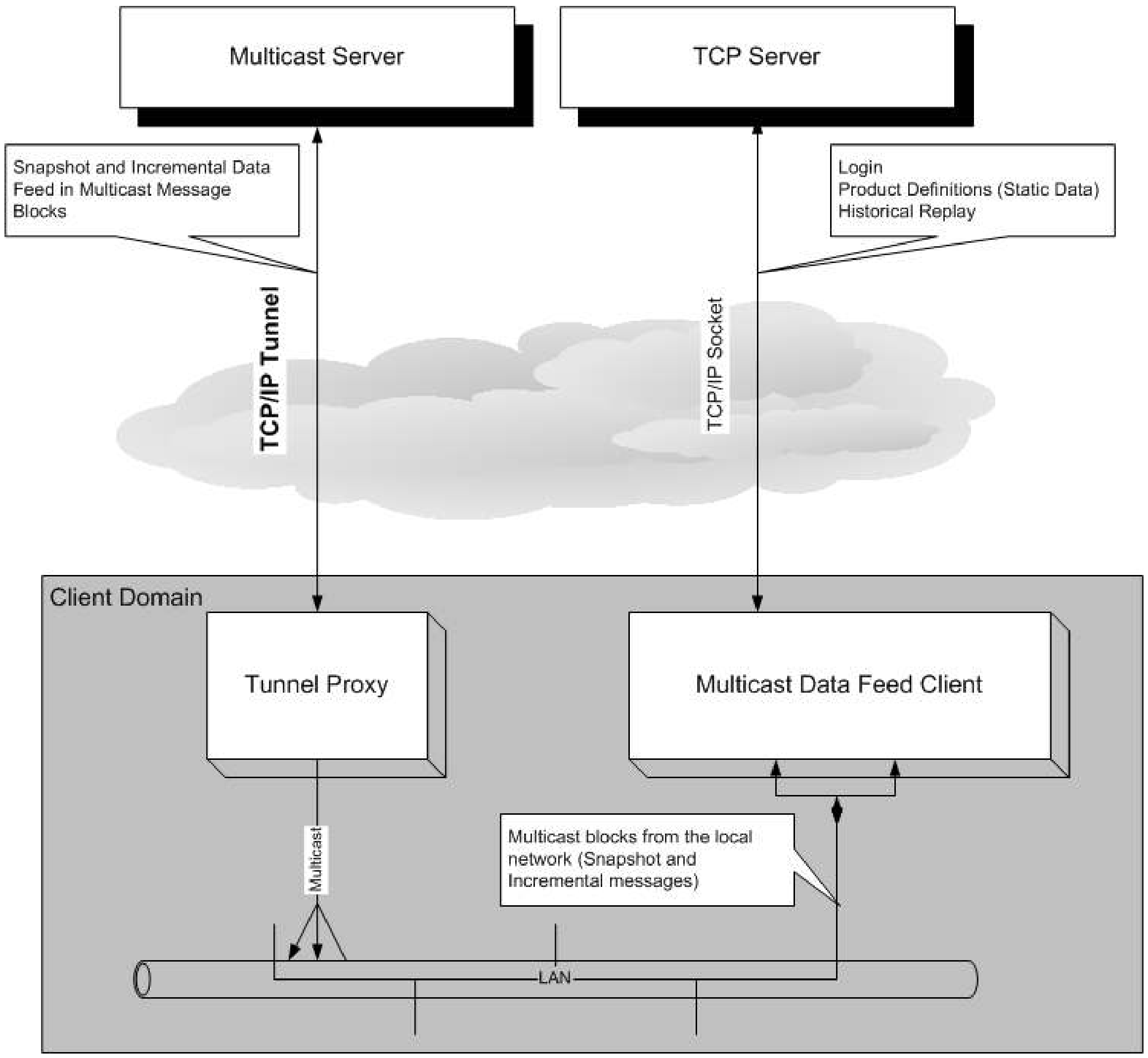
Login is required to connect to the TCP server. If you have test IDs for the iMpact TCP feed already, you can use them again (but not at the same time). Or send us an email (pricefeedsupport@theice.com) if you need new test user IDs.

For multicast traffic, since it cannot be routed through the Internet, you can run a multicast tunnel proxy provided by ICE. Please read the section below for more details.

3. Multicast Tunnel Proxy

For test environment, the multicast tunnel proxy can be used to receive multicast messages through a TCP tunnel. The tunnel proxy establishes a socket connection to the feed server, receives the messages and multicasts them on the local network based on the multicast group mapping rules from TunnelProxy.properties file.

Once the tunnel proxy is launched, multicast client can join the local multicast channels that the tunnel is currently multicasting. This provides a seamless operation from the clients' perspective. The diagram below illustrates the connectivity


Multicast Server
TCP Server
Snapshot and Incremental Data Feed in Multicast Message Blocks
Login
Product Definitions (Static Data)
Historical Replay
TCP/IP Tunnel
TCP/IP Socket
Client Domain
Tunnel Proxy
Multicast Data Feed Client
Multicast
Multicast blocks from the local network (Snapshot and Incremental messages)
LAN

The tunnel TCP server address is 63.247.113.163. But the tunnel port could be different depending on the multicast groups. You can find the port numbers under connection info for API Test env in "ICE iMpact Multicast Feed Technical Specification".

4. Sample Clients

The Getting Started package includes sample test clients along with source code. The following sections provide the details of each of the client applications.

Prior to running any of the client applications, please make sure you do the following:

- Rename all the batch files with .bat extension

- Modify the common.bat file to include your JAVA_HOME and other parameters as applicable for your runtime environment

- For applications that use log4j, modify the log4j.xml as applicable (to set the logging level and other details. If you are going to be using the SMTP appender, you will also have to include the SMTP server, user information (Multicast Monitor uses log4j)

4.1 Simple Multicast Client

This is a simple command line Java client for verifying whether client is getting multicast messages.

What does it do?

1. Joins a multicast group as specified by the following system properties

   multicast.group.address
   multicast.port
   multicast.network.interface

2. Waits for messages from the server, de-serializes them and prints them (debug format, not the binary format) out as they arrive

How to run it?

1. Edit the simpleMulticastClient.bat to setup the desired multicast group address and port numbers. In addition the network interface can be specified for listening to multicast on a specific interface

2. Run “simpleMulticastClient.bat”

4.2 Multicast Monitor Client

This is an extended version of the Simple Multicast Java client used for monitoring the multicast messages through various channels. This is a multi-threaded version that uses log4j to send out email alerts in case of sequence gaps and other error conditions.

What does it do?

1. Spawns multiple threads to join the list of multicast groups as specified by the following system property

   multicast.group.addresses

   For example:

   multicast.group.addresses=233.156.208.251:20005,233.156.208.250:20006

   This is a comma separated list of multicast group information in the format ip:port

   You can set the following property for running in silent mode on or off.

   silent=true (to turn on the silent mode, false for logging all the messages)

   The following system property is used for specifying inactivity threshold. The following sets the threshold to 20 seconds.

   multicast.inactivity.threshold=20000

   In addition, the following property can be used to listen for multicast on a specific network interface

   multicast.network.interface

2. Waits for messages from the server, de-serializes them and prints them (debug format, not the binary format) out as they arrive

3. Sends out alerts in case of errors involving sequence gaps (you will need to configure the SMTP information in log4j.xml for getting email alerts)

How to run it?

1. Edit the multicastMonitorClient.bat to setup the desired multicast group addresses and port numbers
2. If needed, specify the network interface to listen on, using the property multicast.network.interface
3. Run “multicastMonitorClient.bat”

4.3 Simple TCP Client

This is a simple command line Java client for checking connectivity to the TCP feed server, verifying login and whether the client can receive the static product definition messages.

What does it do?

1. Connects to the feed server with information from the simpleTcpClient.properties file
2. Sends login request and product definition request for the interested market types as configured in the properties file
3. Prints the responses on the console in debug format

How to run it?

1. Modify the simpleTcpClient.properties file to use the correct ip/port, user name, password and interested market types
2. Run simpleTcpClient.bat

4.4 Multicast GUI Client

This is a GUI client that illustrates connectivity to the Price Feed server and processing multicast messages. It shows how to synchronize the Snapshot and Incremental channels and maintain the book. It has logic to detect packets with sequence problems.

What does it do?

3. Starts a socket reader thread that connects to the TCP server configured in multicastClientConfig.xml and requests for product definitions.
4. Starts a pair of multicast receiver threads for receiving the snapshot and live messages, handles the synchronization of snapshots and live updates.
5. The consumer threads associated with the receivers dispatch the messages to the appropriate message handlers
6. Each message handler associated with the message types illustrate how to process different types of messages for keeping the book

How to run it?

1. Modify “multicastClientConfig.xml” to use valid username/password.

2. Run “multicastClient.bat”, which will launch a GUI and also connect to the feed server

3. The list of “interested” market types (configured in the multicastClientConfig.xml file) is displayed on the left hand side panel. Choose a market type by left clicking on the market types panel

4. Make sure you have setup the properties in the multicastClientConfig.xml file, for the selected environment such as ‘apitest’ or ‘perftest’.

   Multicast channel context (fullOrderDepth, priceLevel)
   multicastNetworkInterface (if applicable)
   live/snapshot address pairs for the given context

5. Double-click any given market to bring up a book/price level display
6. The status information is logged using the java logging facility, as configured in the *mdflogging.properties* file
7. Please check the log file and watch your console for any errors

Multicast GUI Client - Functionality

The MulticastClient demonstrates how to process the messages from the feed server, and building a book (Full Order Depth/Price Level). It shows how the book is initially constructed using the snapshot messages and updated using order related messages. The sample source code also illustrates how we maintain ordering of price levels for bid/offers by using certain collections.

The multicastClientConfig.xml file contains the runtime parameters needed for this application. Once launched, the MulticastClient uses a thread to establish a TCP/IP Socket connection to the feed server as specified in the multicastClientConfig.xml file. The socket connection is used to send a login request to the server using the credentials in the properties file. After a successful login, the client then requests for the Product Definitions for the interested market types, as configured in the properties file.

Once the static data has been loaded, the client then joins to a pair of multicast channels for a given multicast context. The multicast channel context could be one of PriceLevel, FullOrderDepth. An initial dialog box lets the user choose the specific environment, context and the multicast group.

For example, the following settings tell the client to launch two multicast channel threads for a “PriceLevel” context. One thread will process Price Level Snapshots and the other will process the Incremental/Live PriceLevel messages.

Multicast Channel Context=PriceLevel
Incremental/live multicast address=239.12.255.222
Incremental/live multicast port=5100
Snapshot Multicast Address=239.12.255.222
Snapshot Multicast Port=6100

The client maintains different kinds of books depending on the multicast context. For Full Order Depth, the client maintains and displays each individual order along with aggregated information to the top of book. For Price Level context, only the top 5 levels are maintained.

Multicast Message Synchronization

One of the key functionalities of the multicast GUI client, is its ability to process the multicast feed from two different channels (snapshot/incremental) and synchronize them. There is a pair of threads associated with these two multicast channels. Since the client could be joining the multicast channels any time in the middle of the feed, it uses the session/sequence numbers to find out the order in which the messages have to be processed. Also, it can intelligently determine if a snapshot message is “too” old, and thus have to wait for the next cycle.

At the same time the live/incremental thread could be receiving a lot of messages. These messages are queued until the snapshot has been applied to a specific market. Finally, the live messages are applied, after dropping any messages that are older than the snapshot.

After the snapshot has been loaded for all interested markets, the client then un-joins from the Snapshot multicast channel, and continue to process the live messages.

Failure Conditions

While receiving the multicast feed, clients should be prepared to handle certain failure scenarios such as duplicate messages and sequence gaps.

Sequence problems can occur as a result of a packet drop from the multicast channel, or as a result of the server restarting. In the former case, we detect this condition by the existence of a gap in sequence number. In the latter case, we detect a session number change.

Any time the client detects a sequence problem, one of the following actions are taken, based on what is configured in the properties file.

shutdown - client shuts down
restart - client re-initializes the internal market/book data and starts

Clients should also be prepared to handle duplicate packets. The sample client simply discards duplicate packets.

Requesting Historical Data

The Multicast GUI client also illustrates how to request historical replay of the data feed for a specific session and a sequence range. Select the menu item “Historical Replay” from the “On Demand” menu to request historical data. As illustrated, historical data is requested through TCP/IP socket connection.

Key source files and packages

While reviewing the source code, please pay special attention to the following java classes.

TCP

MDFClientSocketReader.java
MDFClientMessageConsumer.java
MDFClient.java

Multicast

See files under package com.theice.mdf.multicast.*

Functionality (Maintaining Book)

Market.java
MarketOrder.java
PriceLevel.java
AddModifyOrderHandler.java
DeleteOrderHandler.java
TradeMessageHandler.java

Message Classes

The Java source code for the message classes that do serialization and de-serialization are included, serving as examples on how the feed message spec can be implemented in Java. These are part of the java package *com.theice.mdf.message.**

While you can use them as a reference for your implementation, please beware that they won't be supported as a library.

4.5 OS Specific Considerations

This section contains the cause and solution for specific problem areas that we know about.

Situation #1: Getting Connection Exception in Linux

While running on a Linux platforms, you might experience problems while joining a multicast group.

Details

This problem usually shows up as a bind failure while connecting to the multicast socket. A typical error might look like the following.

```
TunnelProxy Starting...
TunnelProxyManager Initializing...
Starting the log manager...
java.net.ConnectException: Connection timed out
at java.net.PlainSocketImpl.socketConnect(Native Method)
at java.net.PlainSocketImpl.doConnect(PlainSocketImpl.java:333)
at java.net.PlainSocketImpl.connectToAddress(PlainSocketImpl.java:195)
at java.net.PlainSocketImpl.connect(PlainSocketImpl.java:182)
at java.net.SocksSocketImpl.connect(SocksSocketImpl.java:366)
at java.net.Socket.connect(Socket.java:518)
at java.net.Socket.connect(Socket.java:468)
at java.net.Socket.<init>(Socket.java:365)
at java.net.Socket.<init>(Socket.java:179)
at
com.theice.mdf.client.multicast.tunnel.TunnelProxy.openTunnel(TunnelProxy.java:46)
at com.theice.mdf.client.multicast.tunnel.TunnelProxy.main(TunnelProxy.java:129)
java.net.ConnectException: Connection timed out
at java.net.PlainSocketImpl.socketConnect(Native Method)
at java.net.PlainSocketImpl.doConnect(PlainSocketImpl.java:333)
at java.net.PlainSocketImpl.connectToAddress(PlainSocketImpl.java:195)
at java.net.PlainSocketImpl.connect(PlainSocketImpl.java:182)
at java.net.SocksSocketImpl.connect(SocksSocketImpl.java:366)
at java.net.Socket.connect(Socket.java:518)
at java.net.Socket.connect(Socket.java:468)
at java.net.Socket.<init>(Socket.java:365)
at java.net.Socket.<init>(Socket.java:179)
at
com.theice.mdf.client.multicast.tunnel.TunnelProxy.openTunnel(TunnelProxy.java:46)
at com.theice.mdf.client.multicast.tunnel.TunnelProxy.main(TunnelProxy.java:129)
```

Solution

This problem is resolved by explicit use of IPv4 stack in Linux. Add the following flag in the java program launch script.

```
-Djava.net.preferIPv4Stack=true
```

Situation #2: Multicast packets leave the box even after explicitly using the loop back (127.0.0.1) interface

While running on a Linux platforms, you might experience problems while launching the tunnel proxy bound to the loop back interface.

Details

This problem usually shows up as the multicast packets leaving the box even after the interface is explicitly set to 127.0.0.1

Solution

This problem is resolved by explicit use of IPv4 stack in Linux. Add the following flag in the java program launch script.

```
-Djava.net.preferIPv4Stack=true
```

5. Support

For technical support, you can do one of the followings

a) Open a ticket through our support ticketing portal http://www.theice.com/integrate . If you don't have the logon credentials, please send a mail to integrate@theice.com. This is preferred channel for support.

b) Alternatively you can write to pricefeedsupport@theice.com