

How to migrate MD application from FA C++ to MD C++ library

Headers

To avoid potential conflicts between FA headers and MD headers in one application we removed *B2BITS_* part from MD headers, instead we put new files into *b2bits* subdirectory.

There is only one exception - header files that belong to a particular handler, e.g. *B2BITS_BovespaApplication.h* remain unchanged.

Namespaces

For the same reason - to avoid potential conflicts between FA and MD libraries - namespaces has also been changed. in MD C++ library there 2 main C++ namespaces:

1. namespace *B2bits* - for common code, not related only to market data, e.g. basic integer types - *u32*, *i32*, etc
FIX related types - *Decimal*, *UTCTimestamp*, etc
2. namespace *B2bits::MD* - for market data related stuff, e.g. *Engine*, *FIXMessage*

Legacy per-handler namespaces remain unchanged, e.g. - *Bovespa*, *Mit*, *Spectra*.

Init, cleanup & basic use cases

MD C++ library uses exactly the same approach FA library does. Instead of *FixEngine* there is *MD::Engine*, which must be initialized first with *init* function, and destroyed with *destroy* method.

MD::Engine provides exactly the same set of methods to create MD handlers FA engine does.

See the examples below, first using *Bovespa* MD handler with FA, seconds - with MD library. The only changes required - different engine C++ header and namespace.

FA C++

```
// assuming include directory - FA-2.24/headers

// FA engine header
#include <B2BITS_FixEngine.h>

// MD handler includes
#include <B2BITS_BovespaApplication.h>
#include <B2BITS_BovespaApplicationListeners.h>

// initialize FIX engine first
Engine::FixEngine::init("engine.properties");

// create MD handler
Bovespa::BovespaApplicationParams params;
auto application = Engine::FixEngine::singleton()->createBovespaApplication( params );

// use application to subscribe and listen to market data
Bovespa::BovespaSubscriptionItem instument;
application->subscribeToInstrument( instument );
...

// release the application
application->release();

// destroy FIX engine
Engine::FixEngine::destroy();
```

MD C++

```
// assuming include directory - Bovespa-2.24/include

// new engine header
#include <b2bits/Engine.h>

// legacy MD handler includes
#include <B2BITS_BovespaApplication.h>
#include <B2BITS_BovespaApplicationListeners.h>

// initialize MD engine first
B2bits::MD::Engine::init("engine.properties");

// create MD handler
Bovespa::BovespaApplicationParams params;
auto application = B2bits::MD::Engine::singleton()->createBovespaApplication( params );

// use application to subscribe and listen to market data
Bovespa::BovespaSubscriptionItem instument;
application->subscribeToInstrument(instument);
...

// release the application
application->release();

// destroy FIX engine
B2bits::MD::Engine::destroy();
```

FIX interface

MD C++ library provides its own FIX interface which is very similar to FIX interface in FA.

FIX interface contains same old FIXMessage, TagValue, FIXGroup classes in B2bits::MD namespace. Main differences are as follows:

- MD FIX interface provides read-only access to messages(getters only).
- No FIXFields and FIXField namespaces, replacement - B2bits::FIXTag
- getEntry and getGroup methods return values instead of pointers:

```
const TagValue getEntry(int index) const;
```

```
const FIXGroup getGroup(int tag) const;
```

see examples below - using FIX interface with FA and MD C++ libraries

FA C++

```
// reading market data snapshot using FA FIX interface
void readSnapshot(const Engine::FIXMessage& msg)
{
    const Engine::FIXGroup& gr = msg.getAsGroup( FIXFields::NoMDEntries );
    for( int i = 0; i < gr.size(); ++i )
    {
        const Engine::TagValue& entry = *gr.getEntry(i); // returns pointer
        switch( gr.getAsChar( FIXFields::MDEntryType, j ) )
        {
            {
            }
        }
    }
}
```

MD C++

```
// reading market data snapshot using MD FIX interface
void readSnapshot(const B2bits::MD::FIXMessage& msg)
{
    using namespace B2bits;

    const MD::FIXGroup& gr = msg.getAsGroup( FIXTag::NoMDEntries );

    for( int i = 0; i < gr.size(); ++i )
    {
        const MD::TagValue& entry = gr.getEntry(j); // returns const value

        switch( gr.getAsChar( FIXTag::MDEntryType, j ) )
        {
        }
    }
}
```

engine.licence

No changes required, FA license will work with MD C++ library