MDP 3.0 - Simple Binary Encoding

MDP 3.0 uses compact Simple Binary Encoding (SBE) optimized for low latency of encoding and decoding while keeping bandwidth utilization reasonably small. Concise message sizes are used but without the processing cost of compression. All FIX semantics are supported. The encoding standard is complimentary to other FIX standards for session protocol and application level behavior.

CME Group has developed SBE in coordination with the FPL Working Group to optimize electronic exchange of financial data, particularly for high volume, low latency data dissemination. This topic describes implementation of SBE in receiving and processing CME Group FIX encoded electronic market data.

Binary Type System

To support traditional FIX semantics, all the documented field types are supported. However, instead of printable character representations of tag-value encoding, the type system binds to native binary data types, and defines derived types as needed.

The binary type system:

- provides a means to specify precision of decimal numbers and timestamps, as well as valid ranges of numbers
- differentiates fixed-length character arrays from variable-length strings
- allows a way to specify the minimum and maximum length of strings that an application can accept
- provides a consistent system of enumerations, Boolean switches and multiple-choice fields.

Binary Coding

MDP Decoder

Real Logic Ltd. and Informatica have collaborated to create open source tools that provide extensive support for Simple Binary Encoding (SBE), the messaging standard developed through the Financial Information Exchange (FIX) Trading Community. The SBE decoders will create an environment that can be used directly by customers or treated as a reference implementation that can be extended into custom solutions tailored to individual customer's needs. The information and tool are open to the public under an Apache Public License and are available here.

The FPL Simple Binary Encoding Specification is available here.

Users must be registered with FIX Protocol Organization to access the Simple Binary Encoding Documentation and will need a User ID provided by FPL to logon to the site.

FPL Simple Binary Encoding

The key features of Simple Binary Encoding are:

- Little-endian byte ordering
- Based on IEEE primitive data type encodings
- Extension with “null” values
- Complex data types are broken down with primitive data types
  - Primitive integers are native types on popular platforms (avoids character conversion)
  - Decimal numbers for prices have flexible or fixed precision
  - Timestamps, times and dates are sent as numeric units of time rather than verbose strings
  - Enumerations are well defined
- Maximizes direct access
- High flexibility
- Very low latency
- Low encoding/decoding processing
- Message schema-based to drive content
- Uses templates to indicate which operations to use in decoding

Simple Binary Encoding supports traditional FIX field types without conversions. Direct data access is provided without complex transformations or conditional logic. This is achieved by:

- Usage of native binary data types and simple types derived from native binaries, such as prices and timestamps.
- Preference for fixed positions and fixed length fields, supporting direct access to data and avoiding the need for management of variable-length elements which must be sequentially processed.
- Primitive integers are native types on popular platforms, avoids character conversion
- Decimal numbers for prices have flexible or fixed precision
- Timestamps, times and dates are sent as numeric units of time rather than verbose strings

- Enumerations are well defined
- Multi-value choice as a bitset

Client systems have the flexibility to implement the standard in a way that best suits their needs.

Binary Encoding Example

The following timestamp:

> UTC timestamp 14:17:22 Friday, October 4, 2024

is expressed in binary code (nanoseconds since Unix epoch) this way:

> 007420bf8838fb17 (8 bytes in nanoseconds since Unix epoch synced to a master clock to microsecond accuracy.

and expressed in tag-value encoding this way

```
20241004 14:17:22.000 (21 bytes with millisecond precision)
```