Warsaw Automated Trading System

NAR-181 BenDec Message Definition Format

Przeprowadzenie prac badawczo-rozwojowych w zakresie opracowania oraz implementacji zintegrowanej nowoczesnej Platformy Transakcyjnej przełomowej pod względem parametrów wydajnościowych i pojemnościowych oraz nowatorskich protokołów komunikacyjnych oraz algorytmów zawierania transakcji

POIR.01.01.01.-00.0829/18-00

date: 2021-03-25

version: 3.0.1

1. CONTENTS

2. DISCLAIMER

This document is for information purposes only. Reasonable care has been taken to ensure the details contained here are accurate and not misleading at the time of preparation. WSE is not responsible for any errors or omissions contained in this document.

WSE reserves the right to treat specifications contained in this document as the subject to the later change without prior notice.

This document contains information confidential and proprietary to Warsaw Stock Exchange, and may not be reproduced, disclosed or used in whole or part, in any manner, without the express prior written consent of WSE.

This document is part of the WATS project documentation and is significantly modified during the development process.

3. PREFACE

3.1. TARGET AUDIENCE

This document is dedicated to entities producing software compatible with the WATS.

4. NAR-181 DOCUMENT PURPOSE

This document contains BenDec Message format description. BenDec is format used to define Binary Trading Port and Market Data messages by the WATS.

5. NAR-181 BENDEC MESSAGE DEFINITION FORMAT

All messages used in the system are defined using the custom BenDec (Binary Encoder Decoder) format. It uses JSON files for the artifacts used to describe message structure.

The BenDec description format is used to define the binary (BIN) and market data (MD) protocols. The FIX protocol uses standards defined by the FIX Community.

The BenDec (https://github.com/fudini/bendec) library currently supports the Rust, TypeScript, and C++ languages.

Protocol definitions are organized into 3 different files:

- shared.json – contains definitions shared by all protocols;
- md.json – contains the definition for the Market Data Protocol (MD);
- trading_port.json - contains the definition of the Binary Protocol (BIN).

5.1. MESSAGE UNION

The main part of the definition is the special union `Message`, which enumerates all structures used as messages.

```
{
  "kind": "Union",
  "name": "Message",
  "members": [
    "Heartbeat",
[...]
  ],
  "discriminator": [
  "header",
  "msgType"
]
```

As shown above, the type of the message is determined using the `msgType` field in the header field.

5.2. MSGTYPE ENUMERATION

Each message has a `msgType` value assigned, listed in the `MsgType` enumeration.

```
{
  "kind": "Enum",
  "name": "MsgType",
  "underlying": "u16",
  "variants": [
    [
```

```
      "Heartbeat",
      1,
      "A message type used to check connectivity."
    ],
[...]
  ]
},
```

Each message is described as a JSON array:

1. Name of the structure used as a message;
2. `msgType` value assigned to the message;
3. Message comment.

5.3. PRIMITIVE TYPES

Each primitive language-defined type is defined using the Primitive keyword.

```
{
  "kind": "Primitive",
  "name": "u16",
  "size": 2,
  "description": "Unsigned integer, 16-bit."
},
```

A primitive declaration has the following fields:

1. kind == "Primitive";
2. name – name of the primitive type;
3. size – size of the type (in bytes);
4. description – comment on the primitive.

5.4. ALIASES

Aliases are used for defining domain-related names for other types.

```
{
  "kind": "Alias",
  "name": "Date",
  "alias": "u32",
  "description": "Date (yyyymmdd) as integer value.\r\nIn case
of undefined date value of '0' (zero) is used."
},
```

An alias definition has the following fields:

1. kind == "Alias";
2. name – name of the defined alias type;
3. alias – name of the aliased type;
4. description – comment on the alias.

5.5. ARRAYS

An array is used to define a simple array of items.

```
{
  "kind": "Array",
  "name": "BICCode",
  "type": "AnsiChar",
  "length": 8,
  "description": "Business Identification Code as specified in ISO 9362."
},
```

An array definition has the following fields:

1. kind == "Array";
2. name – name of the defined type;
3. type – type of items;
4. length – number of items;
5. description – comment on the array.

5.6. ENUMERATIONS

Enumerations are defined as follows:

```
{
  "kind": "Enum",
  "name": "IssueSizeType",
  "description": "Type of issue size.",
  "underlying": "u8",
  "variants": [
    [
      "Quantity",
      1,
      "Issue size expressed in quantity."
    ],
    [
      "Value",
```

```
      2,
      "Issue size expressed as nominal value."
    ],
    [
      "NoIssueSize",
      3,
      "No issue size."
    ]
  ]
},
```

An enumeration definition uses the following fields:

1. kind == "Enum";
2. name – name of the enumeration;
3. description – comment on the enumeration;
4. underlying – name of type that is used for values;
5. variants – array of elements:
   a. name – enumerated value name,
   b. value – value of enumerated value,
   c. comment – comment on the value.

5.7. STRUCTURES

A structure (struct) is a set of fields grouped for convenience.

```
{
  "kind": "Struct",
  "name": "Header",
  "description": "Header used in Core Bus messages.",
  "fields": [
    {
      "name": "length",
      "type": "MsgLength",
      "description": "Total length of the message."
    },
[...]
  ]
},
```

A structure definition has the following fields:

1. kind == "Struct";
2. name – name of the structure;
3. description – comment on the structure;
4. fields – array of attributes:
    a. name – name of the attribute,
    b. type – type of the attribute,
    c. description – comment on the attribute.

6. PROTOCOL DEFINITION – SHARED STRUCTURES

The shared part of BenDec definitions should be saved to the shared.json file.

6.1. PRIMITIVES

Name	Size	Note
u8	1	Unsigned integer, 8-bit.
u16	2	Unsigned integer, 16-bit.
u32	4	Unsigned integer, 32-bit.
u64	8	Unsigned integer, 64-bit.
i64	8	Signed integer, 64-bit.
f64	8	A 64-bit floating point type (specifically, the “binary64” type defined in IEEE 754-2008).

6.2. ALIASES

Name	Basic type	Note
AnsiChar	u8	String.
BusinessClassificationId	u32	ID of the business classification.
ClientId	u16	ID of the client.
ClientOrderId	u32	ID of the client order.
Date	u32	Date (yyyymmdd) as integer value. In case of undefined date value of '0' (zero) is used.
IssuerId	u32	ID of an issuer.
MarketOperatorId	u32	ID of the market operator.
MarketSegmentId	u32	ID of the market segment.
MarketSegmentTypeId	u32	ID of the market segment type.
MsgLength	u16	Length of the message.
MsgVersion	u16	Message version - used to differentiate between messages sent using different versions of the protocol.
OrderCount	u16	---
OrderId	u64	ID of a given order.
Port	u16	Port number.

Name	Basic type	Note
Price	i64	Price of the product.
ProductId	u32	ID of the product.
ProductIdentificationTypeId	u32	ID of the product identification type (for example ISIN or SEDOL).
ProductIssueSize	f64	Issue size of the product.
ProductNominalValue	f64	Nominal value of the product.
ProductTypeId	u32	ID of the product type.
Quantity	u64	Quantity of a tradable product.
SeqNum	u32	Sequence number of the message.
ServiceId	u16	ID of the service (system component).
SessionId	u16	ID of the session.
Timestamp	u64	Timestamp (exact date and time of an event).
TradableProductId	u32	ID of the tradable product.
TradeId	u32	ID of the trade (match between buy and sell orders).
TradingConfigId	u32	ID of the trading configuration.
TradingGroupId	u32	ID of the trading group.
TradingVenueId	u32	ID of the trading venue.

6.3. ARRAYS

Name	Type	Note
BICCode	AnsiChar[8]	Business Identification Code as specified in ISO 9362.
BusinessClassificationCode	AnsiChar[10]	Code of the business classification.
BusinessClassificationName	AnsiChar[50]	Name of the business classification.
CFICode	AnsiChar[6]	Clarification of Financial Instruments as specified in ISO 18774.
CountryCode	AnsiChar[3]	Three-letter country code.
CountryId	AnsiChar[2]	Short country code, for example "PL".
CountryName	AnsiChar[50]	Country name, for example "Poland".
CurrencyId	AnsiChar[3]	Three-letter ID of a given currency (e.g. USD).
CurrencyName	AnsiChar[50]	Name of the currency, for example "Pound sterling".
FISNCode	AnsiChar[35]	Financial Instrument Short Name as specified in ISO 10962.
Hostname	AnsiChar[32]	DNS host name.

Name	Type	Note
IssuerName	AnsiChar[50]	Name of the product's issuer.
LEICode	AnsiChar[20]	Legal Entity Identifier as specified in ISO 17422.
MarketOperatorName	AnsiChar[50]	Name of the market operator.
MarketSegmentTypeName	AnsiChar[50]	Name of the type of market segment.
MICCode	AnsiChar[4]	Market Identifier Code (MIC) as specified in ISO 10383.
ProductIdentification	AnsiChar[50]	Product identification, for example its ISIN number.
ProductIdentificationTypeName	AnsiChar[10]	Name of the given product identification type.
ProductName	AnsiChar[50]	Name of the product.
ProductTypeCode	AnsiChar[10]	Code of the product type.
ProductTypeName	AnsiChar[50]	Name of the product type.
TestPayload	u8[1436]	This is the largest UDP packet the system can work with. The length is 1500 minus message header size (36 b) minus IP and UDP header sizes (28 b) = 1436.
TextMessage	AnsiChar[50]	A text message.
Token	AnsiChar[8]	Unique token.
TradingConfigName	AnsiChar[50]	Name of the trading configuration.
TradingGroupName	AnsiChar[50]	Name of the trading group.
TradingVenueName	AnsiChar[50]	Name of the trading venue.

6.4. ENUMS

Name	Type	Notes
IssueSizeType	u8	Type of issue size.
NominalValueType	u8	Indicates the security's nominal value type.
OrderSide	u8	Indicates order side (buy or sell).
PriorityFlag	u8	Indicates whether the priority flag was lost or retained.

IssueSizeType enumeration

Type of issue size.

Name	Value	Notes
Quantity	1	Issue size expressed in quantity.
Value	2	Issue size expressed as nominal value.
NoIssueSize	3	No issue size.

NominalValueType enumeration

Indicates the security's nominal value type.

Name	Value	Notes
NoNominal	1	Indicates that the security has no nominal value.
Constant	2	Indicates that the security has a constant nominal value.
Unknown	3	Indicates that the security has an unknown nominal value.

OrderSide enumeration

Indicates order side (buy or sell).

Name	Value	Notes
Buy	1	Indicates a buy-side order.
Sell	2	Indicates a sell-side order.

PriorityFlag enumeration

Indicates whether the priority flag was lost or retained.

Name	Value	Notes
Lost	1	The priority flag was lost.
Retained	2	The priority flag was retained.

6.5. STRUCTS

Currently, there are no shared structs defined.

6.6. MESSAGES

Because it is required for a `msgType` value to be included in the `MsgType` enum, no messages are defined as shared.