Document title

SBE TEMPLATE INTRODUCTION AND COMPATIBILITY RULES

Document type or subject

Technical Notice

Version number
1.0

Date
11 December 2018

Number of pages
13

Author
Euronext

CONTENTS

1. INTRODUCTION TO SBE (SIMPLE BINARY ENCODING)

Scope and audience: This technical note intends to provide general technical information about SBE and specify the SBE implementation on Optiq®. For the SBE integration and specificities of the SBE messages within Euronext Optiq platform please consult the Optiq specifications **HERE**

This document is meant to familiarize developers who will work with SBE for the first time and remind others the current rules of implementation.

SBE stands for Simple Binary Encoding and it is a OSI layer 6 presentation for encoding and decoding binary application messages with the main purpose of supporting low-latency financial applications.
SBE is a FIX standard for binary message encoding: FIX Simple Binary Encoding.

SBE provides a language independent type system supporting integers, floating point numbers, characters, arrays, constants, enums, bitsets, composites, grouped structures that repeat, and
variable length strings and blobs.

SBE has been chosen for the Optiq platform for the enumerated advantages mentioned in the following section.

1.1 ADVANTAGE OF USING SBE

- **Standard:** FIX SBE has been developed by the FIX trading community, under the "High Performance FIX" initiative
- **Performance:** Fast and compact encoding / decoding of FIX messages
- **Simplicity:** Well documented and easy to use, with an open source implementation available
- **Technology adoption:** Major exchanges already use SBE.
- **Efficiency:** Focus on reducing bandwidth utilization for the market data SBE offers the possibility to have backward and forward compatibility. **It means that clients are not required to be on the last version of Schema Version (message structure version) to be able to read the message.** This is only possible if changes to SBE messages that are being introduced are both:
    - Flagged as Optional fields in the SBE template.
    - done at the end of the block and/or the repeating section.

  Euronext will provide explicit communication via Info Flash or CTSG Notice on changes when upgrading templates.
- Using the version of the schema system will parse the known fields and skip anything unknown.

1.2 SBE USAGE IN PRACTICE

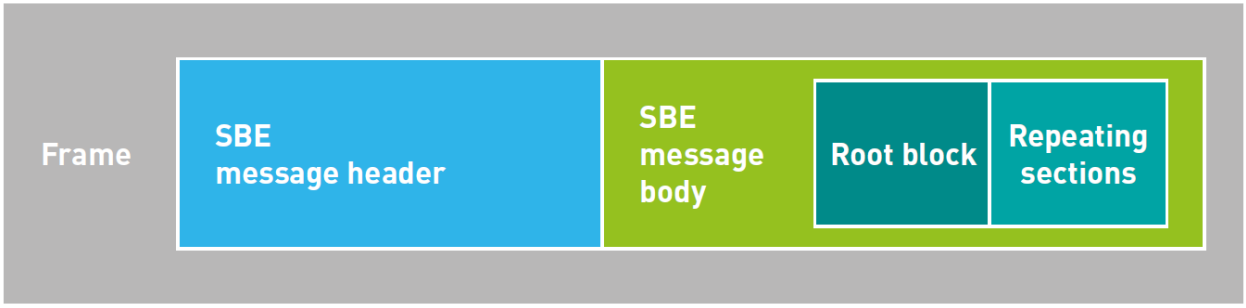
Messages to be encoded and decoded with SBE are described in a schema in XML format. The XML schema is a machine readable version of the order-entry and market data message specifications.
The XML message schema can be used for automated code generation. An SBE-compatible code generator can use the XML definition as an input and generate encoding/decoding methods in a high level language like Java, C++ or C#.

While using such automated code generation is not mandatory to use FIX SBE, it is expected to significantly reduce development effort on the long run:

- Elimination of human errors introduced by manual coding.
- Changes to message definition can be directly fed into a continuous integration process to test for impact and regressions.
- An example of such code generation tool-chain has been developed as an open sourced project by [Real Logic] (http://real-logic.co.uk). It is available HERE. Real Logic open sourced code generator is a Java TM tool and currently supports Java, C++, and C# as target language's.

1.3 SBE MESSAGE STRUCTURE

A general representation of a SBE message structure is provided below.



- Frame – contains the total length of the message, including length of the Frame and SBE Header fields
- SBE Message Header - contains general information about the SBE template used, specifically about Block length, Template ID, Schema ID, Schema version
- SBE Message Body – contains the list of static fields of the root block and the fields of the repeating sections. An SBE Message body may have none, one or multiple repeating sections depending on the specificities of the message. Among others fields, the root block of each repeating section contains two bytes, NumInGroup and BlockLength that respectively describe the number of occurrences of the repeating section and the size of each.
- The standard SBE uses only two types of presence: Mandatory or Optional, the mandatory field being the default value. In Euronext OEG technical specifications the presence of a field can be also conditional however this type of field is represented in the SBE template as Optional field and the conditionality is managed by the Matching Engine. To guarantee the backward and forward compatibility, all new fields on market data and order entry that will be added will be optional in the SBE Template. Then, if an order entry field is required for a new service or a new functionality, the

check on the presence of the field and the validation of the format and values will be done in the Matching Engine. If the field is not present or if the format or the content are incorrect then the message will be rejected.

Note: The above structure applies for inbound (OEG) and outbound (OEG, MDG) messages. As the domain of values of a particular field can be extended at any time, to ensure the forward and backward compatibility on customers' applications, we encourage customers to ignore any unknown value or even discard the message depending on the applications' needs.

Technical note for developers: for backward and forward compatibility purposes, customers must comply with the following when reading messages:

- The size of the **root block** is not fixed and is determined by the BlockLength of the SBE message header. Therefore, to jump from the end of the SBE message header to the first repeating section, customers must jump of "BlockLength" bytes.
- The size of **repeating sections** is not fixed and is determined by the NumInGroup and BlockLength of the header of repeating sections. Therefore to jump from the end of the header of a given repeating section to the next repeating section, customers must jump of "NumInGroup x BlockLength" bytes.

Please check the standard **FIX SBE** for a general overview of the SBE Message Structure – please note, access to this page requires registration.

Use the Optiq MDG and OEG specifications **HERE**, for more specific information about OPTIQ SBE message structure.

1.4 SBE XML SCHEMA AND OPTIQ INTEGRATION

The SBE XML schema is the template XML file used by the SBE encoder / decoder to encode or decode SBE messages. The SBE XML schema, matching the specification, is available for download on the Euronext File Server (EFS).

Note: The SBE Templates are available in our Euronext file server per segment and the SBE versioning from one segment to another can be different. The naming convention is the following one : OptiqMDG_p-EUA_SBETemplate_Equities_YYYYMMDD.xml

- Availability of SBE templates in Production Environment : 2 days minimum before go live
- Availability of SBE templates in EUA Environment : the day of project implementation

Use the file services user guide **HERE**, for more specific information about how to access Euronext file services EFS

Use the Optiq files specifications **HERE**, for more specific information about SBE files name convention

There are several elements which build an SBE XML schema:

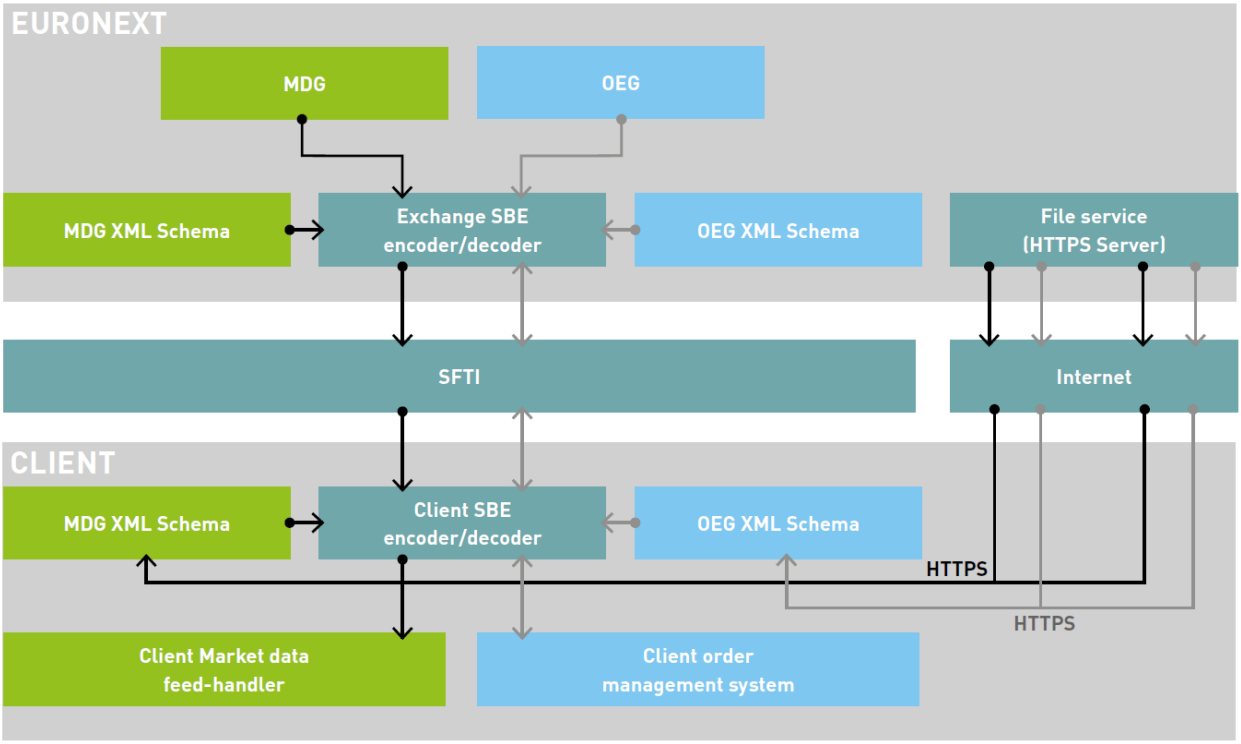
- Message schema, Types (type, set, enum, composite), Message, Field, Group, Data.

Each of the above mentioned elements are being described HERE.

An example of such general SBE schema can be found HERE.

Below, specific to Euronext and its clients, is a diagram of the usage and integration of the SBE Encoder and SBE XML template (schema) within MDG (Market Data Gateway) and OEG (Order Entry Gateway).

This diagram applies to the Optiq Production Environment.



Notes to the diagram:

- Exchange and Client SBE Encoder / Decoder are different.
- MDG and OEG XML schema are the same.

Optiq SBE XML template files for both MDG and OEG will be made available for download on the Optiq File Service.

2. BACKWARD AND FORWARD COMPATIBILITY RULES

2.1 SCHEMA VERSIONING

SBE template for OEG and MDG contains two fields used for version purposes:

- **Version:** This field is used to track changes to a message schema and increments every time a significant change in the SBE is made (example new fields/deprecated fields/new messages etc.) Version is a snapshot in time and all elements in a given generation of the schema share the same version number.
    - A change of SBE version because of a modification on the order entry side implies an upgrade of SBE version on the market data side even if no change is introduced. The opposite is also true. The OEG and MDG SBE templates of a segment are always aligned.
- **semanticVersion:** Now and for the time being, this field is used to identify the OEG or MDG functional specification version the SBE template is associated to. Later on, this field will be reserved for internal use only. Therefore we recommend customers not to use this field. To link the SBE template version mentioned in the functional specification, customers are invited to look at the SBE Version field (see above) in the SBE Template file.
    - Hence a modification of SBE template implies an upgrade of the SBE version and consequently a change of specifications and an upgrade of semanticVersion. However a change of SemanticVersion doesn't necessarily mean an upgrade of SBE Version. It is to be noted that the SBE version is not necessarily sequential. From one version to another one there can be a gap of one or several numbers.
    - Example of a gap:

```
<sbe:messageSchema xmlns:sbe="http://fixprotocol.io/2016/sbe"
xmlns:enx="http://www.euronext.com/dd" xmlns:str="http://exslt.org/strings"
xmlns:ext="http://exslt.org/common" package="com.euronext.optiq.dd" id="0" version="105"
semanticVersion="1.4.0" description="OEG Binary" byteOrder="littleEndian">

<sbe:messageSchema xmlns:sbe="http://fixprotocol.io/2016/sbe"
xmlns:enx="http://www.euronext.com/dd" xmlns:str="http://exslt.org/strings"
xmlns:ext="http://exslt.org/common" package="com.euronext.optiq.dd" id="0" version="107"
semanticVersion="1.5.0" description="OEG Binary" byteOrder="littleEndian">
```

2.2 SBE TEMPLATE MINIMAL VERSION

For retro compatibility reasons, Euronext guarantees it will support several versions of SBE in parallel in the various environments: pEUA, vEUA and PROD. However a SBE template minimal supported version number will be set and revised regularly. Customers will be invited to migrate before a defined date to this SBE minimal version or a higher SBE template version. Messages with a format corresponding to a version smaller than the minimal version SBE will not be accepted on inbound messages. Euronext will send outbound messages with version number corresponding to the version of the current SBE template in use for this segment. Then at the same time the deprecated fields on the outbound messages linked to the SBE Templates versions earlier than the SBE minimal version will be filled with Null values. Customers will be

notified at least 3 months in advance via Info Flash or CTSG Notice to allow them to smoothly upgrade their SBE template.

Once in a while, a major change of SBE that will not be backward and forward compatible will be released. In that case customers will be notified 4 months in advance in order for them to prepare for the change. Through this type of release, all deprecated fields will be removed.

2.3 CLIENT SBE TEMPLATE UPGRADE STRATEGY

SBE offers the possibility to have backward and forward compatibility. **It means that clients are not required to be on the last version of Schema Version (message structure version) to be able to read the message, unless explicitly mentioned in a dedicated communication sent at least with a 4 month notice. Indeed, any SBE template version N contains all specifications contained in version N-1 or earlier plus additional changes, allowing clients to choose when they want to proceed to updates of the SBE templates.**

For example one or more segment on Euronext market can be configured with SBE version 107 while all the other segments will be configured with SBE version 105. In this situation:

- A client using a SBE template version 105 can remain with the same version to talk to every single segment on the cash market **BUT** will not be able to use the latest functionalities introduced in SBE version 107. Therefore clients requiring to use the new functionalities provided in SBE version 107 must upgrade to the latest SBE version. The private messages sent by OEG back to the client as well as public messages sent by MDG will follow the structure corresponding to the SBE template in use in this segment. As a consequence, clients must also make sure that their systems are able to manage and read messages that do not correspond to the SBE template they use (leveraging the backward compatibility of the Euronext change).
- A client using SBE template version 107 can talk to every single segment on the cash market, even if some segments are configured with a lower SBE template version. The private messages sent by OEG back to the client as well as public messages sent by MDG will follow the structure corresponding to the SBE template in use in this segment. As a consequence, clients must also make sure that their systems are able to manage and read messages that do not correspond to the SBE template they use (leveraging the forward compatibility of the Euronext change).

Note: Customers must still comply with the minimal SBE template version described in section 2.2

2.4 ADDING NEW FIELDS

An SBE message body is composed of the below two sections:

- **Root Block:** This section contains static mandatory and/or optional fields. SBE Backward and forward compatibility allows to introduce new fields at the end of the root block sections.
    - Note to developers: The position of the start of the repeating sections is to be identified by using the 'BlockLength' information provided in the SBE message header (see section 1.3 : SBE message structure).

- **Repeating sections:** This section contains mandatory and optional fields**,** similar to the root block section SBE backward and forward compatibility allows to introduce new fields at the end of the **repeating sections**.
    - Note to developers: The position of the start of the next repeating section is to be identified by using the 'NumInGroup' and 'BlockLength' information provided in the SBE repeating section header (see section 1.3 : SBE message structure).

Fields introduced for both root and/or repeating sections are **new optional fields only**.

Note: Some new fields may be flagged in our OEG technical specifications as mandatory or conditional fields, however in the SBE template those fields will be always added as Optional fields. The conditionality or obligation will be handle at the Matching Engine level (see section 1.3).

2.5 UNIQUE IDENTIFIERS

The below identifiers are available in the OEG or MDG SBE templates:

SBE Message Header

- **Message Schema ID (Schema ID):** Currently this "id" is currently set up a with a value equal to "0". This is true for the current Cash implementation and will be also used for the derivatives migration. In the event where backward and forward compatibility mechanisms were to be impacted by a change, Euronext would increment this identifier.

- **Message ID (Template ID):** This identifier is unique per message per Schema ID on the SBE template and follows the same message id described in the Euronext SBE technical specifications document **here.**

    - Example**:** On the Optiq OEG SBE technical specifications the "New Order" message has an id equal to "01" this Id is also used in the OEG SBE template. <sbe:message name="NewOrder" id="1">

SBE Message Body

- **Field ID (root section):** This is a sequential identifier unique per message starting always with "1", this means that the same field can hold a completely different ID between messages. However the field name remains unique throughout the various SBE template versions.
    - Euronext strongly recommends clients **do not** use the field ID as reference to identify unique fields. Instead we recommend to use the name of the field which remains unique through the various types of message and also through the various SBE template versions **Note: customers must be aware that field names are case sensitive**

    - Example: In the "New Order" message the field Symbol index holds an id equals to "5" while in the "Ack" message the same field hold an Id equals to "12"

```
<sbe:message name="NewOrder" id="1">
  <!--Expanding sub message-->
  <field id="1" name="clMsgSeqNum" type="uint32_t"/>
  <field id="2" name="firmID" type="char8"/>
  <field id="3" name="sendingTime" type="uint64_t"/>
  <field id="4" name="clientOrderID" type="int64_t"/>
  <field id="5" name="symbolIndex" type="uint32_t"/>
  <field id="6" name="eMM" type="EMM_enum"/>
  <field id="7" name="orderSide" type="OrderSide_enum"/>
  <field id="8" name="orderType" type="OrderType_enum"/>
  <field id="9" name="timeInForce" type="TimeInForce_enum"/>
  <field id="10" name="orderPx" presence="optional" type="int64_t"/>
  <field id="11" name="orderQty" type="uint64_t"/>
  <field id="12" name="executionWithinFirmShortCode" type="int32_t"/>
```

```
<sbe:message name="Ack" id="3">
  <!--Expanding sub message-->
  <field id="1" name="msgSeqNum" type="uint32_t"/>
  <field id="2" name="firmID" type="char8"/>
  <field id="3" name="sendingTime" presence="optional" type="uint64_t"/>
  <field id="4" name="oEGINFromMember" presence="optional" type="uint64_t"/>
  <field id="5" name="oEGOUTTimeToME" presence="optional" type="uint64_t"/>
  <field id="6" name="bookIn" type="uint64_t"/>
  <field id="7" name="bookOUTTime" presence="optional" type="uint64_t"/>
  <field id="8" name="oEGINFromME" presence="optional" type="uint64_t"/>
  <field id="9" name="oEGOUTToMember" presence="optional" type="uint64_t"/>
  <field id="10" name="clientOrderID" presence="optional" type="int64_t"/>
  <field id="11" name="origClientOrderID" presence="optional" type="int64_t"/>
  <field id="12" name="symbolIndex" type="uint32_t"/>
```

- Note for developers: please refer to section 1.3 for details on how to read the message structure while ensuring backward and forward compatibility of your application.

- **Group ID (repeating section):** This is a sequential identifier **but only unique within a SBE template and message** The same groups Id can be found for two different repeating sections in two versions of the SBE templates if an additional field ID has been integrated into the root section.

  Example: In the SBE template **103** the standing data message, the group ID for **EMMPatternRep** had a value of "58" however in template id **105** the group ID for **EMMPatternRep** had a value of "59" following an addition of a new Field named ICBCode at the end of the root section.

```
<field id="55" name="strikePriceDecimals" presence="optional" type="unsigned_char"/>
<field id="56" name="liquidInstrumentIndicator" presence="optional" type="unsigned_char"/>
<field id="57" name="marketOfReferenceMIC" presence="optional" type="char4"/>
<group dimensionType="groupSizeEncoding" id="58" name="EMMPatternRep">
  <field id="1" name="eMM" type="EMM_enum"/>
  <field id="2" name="patternID" presence="optional" type="uint16_t"/>
  <field id="3" name="tickSizeIndexID" presence="optional" type="uint16_t"/>
  <field id="4" name="marketModel" presence="optional" type="MarketModel_enum"/>
</group>
```

```
<field id="55" name="strikePriceDecimals" presence="optional" type="unsigned_char"/>
<field id="56" name="liquidInstrumentIndicator" presence="optional" type="unsigned_char"/>
<field id="57" name="marketOfReferenceMIC" presence="optional" type="char4"/>
<field id="58" name="iCBCode" presence="optional" type="char8" sinceVersion="104"/>
<group dimensionType="groupSizeEncoding" id="59" name="EMMPatternRep">
  <field id="1" name="eMM" type="EMM_enum"/>
  <field id="2" name="patternID" presence="optional" type="uint16_t"/>
  <field id="3" name="tickSizeIndexID" presence="optional" type="uint16_t"/>
  <field id="4" name="marketModel" presence="optional" type="MarketModel_enum"/>
</group>
```

- **Field ID (repeating section):** This is a sequential identifier unique per repeating section starting always with "1". this means that the same field can hold a completely different ID between messages. Therefore Euronext strongly recommends clients **do not** use the field ID as reference to identify unique fields. Instead we recommend to use the name of the field which remains unique through the various types of message and also through the various SBE template versions.

    - Example: In the "StandingData" message the field Symbol index holds an id equals to "3" while in the "Time Table" message the same field hold an Id equals to "5"

```
<sbe:message name="StandingData" id="1007">
  <field id="1" name="mDSeqNum" type="uint64_t"/>
  <field id="2" name="rebroadcastIndicator" type="unsigned_char"/>
  <field id="3" name="symbolIndex" type="uint32_t"/>
  <field id="4" name="optiqSegment" type="OptiqSegment_enum"/>
  <field id="5" name="partitionID" type="uint16_t"/>
```

```
<sbe:message name="Timetable" id="1006">
  <field id="1" name="mDSeqNum" type="uint64_t"/>
  <field id="2" name="rebroadcastIndicator" type="unsigned_char"/>
  <field id="3" name="eMM" presence="optional" type="EMM_enum"/>
  <field id="4" name="patternID" presence="optional" type="uint16_t"/>
  <field id="5" name="symbolIndex" presence="optional" type="uint32_t"/>
```

2.6 ENCODING TYPES

Encoding tells how a field of a specific data type is encoded on the wire. An encoding maps a SBE data type to either a simple, primitive data type, such as a 32 bit signed integer, or to a composite type.

The encoding types used by Euronext in the SBE template are not expected to be changed through their entire existence. Note that many fields may share a data type and an encoding. In the future though, Euronext may create new encoding types for some of the new fields which would be created with a newer version.

2.7 EXTENSION OF VALUES IN EXISTING FIELDS

Inbound messages : there cannot be any deletion of authorized values for any enum or bitset type of fields to keep the backward forward compatibility. If a service becomes obsolete or has been decommissioned then the value will be deprecated. If for some reason a value needs to be deleted, the backward and forward compatibility will be broken and therefore enough time will be provided so customers can upgrade their application (please refer to section 2.9).

If any authorized value is added, it is only valid/supported on the newly introduced feature or service (i.e. it will only be supported by segments offering this new feature or service).

Outbound messages: there cannot be any deletion of authorized values for any enum or bitset type of fields to keep the backward forward compatibility. If a service becomes obsolete or has been decommissioned then the value will be deprecated. If for some reason a value needs to be deleted, the backward and forward compatibility will be broken and therefore enough time will be provided so customers can upgrade their application (please refer to section 2.9).
If customers don't intend to upgrade to a new version of SBE template, to ensure backward compatibility, customers must expect and properly ignore any current unknown authorized value on enum and bitset type of fields.

2.8 DEPRECATED VALUES, FIELDS & MESSAGES

For a given SchemaID, deprecated fields are part of the backward and forward compatibility and allow customers to easily identify fields at any level (root or repeating sections) that should no longer be used in new messages. A version number is associated with the deprecated field to notify from which SBE Template version this field is deprecated.

To guarantee backward and forward compatibility, the following rules apply:

Deprecated Value	IN (if the customer continues to use it)	The message is processed or rejected depending on the business importance of the field (see specifications).
	OUT	Deprecated values are not sent anymore.
Deprecated Field	IN (if the customer continues to use it)	The message is processed or rejected depending on the business importance of the field (see specifications)*.
	OUT	Customers must accept and ignore Null_Value*.
Deprecated Message	IN (if the customer continues to use it)	The message will be rejected.
	OUT	Deprecated messages will not be sent anymore.

IN = Msg from Client to Optiq (OEG) / OUT = Msg from Optiq to Client (OEG/MDG)
*Two enum fields are exceptionally deprecated in SBE Template 107 as they are not currently used by customers and will not be used in the future.

2.9 SBE RELEASES RULES

Break of backward and forward compatibility rules

On a very exceptional basis, Euronext could perform 'cleaning' of fields. Similarly, Euronext reserves the right to break forward and backward compatibility in exceptional circumstances and not follow the rules mentioned above. In that cases:

- Notice will be given at least 4 months in advance
- SchemaID will be incremented
- Clients would be asked to do mandatory conformance testing

Note: Please refer to section 2.2. SBE template minimal version.

SBE template releases outside a project

Euronext can, at any time, release a newer version of the SBE template on any given segment. This will mostly happen in two cases:

- Euronext upgrades one or several systems on the trading platform. There is no direct impact on external customers but to ensure compatibility on the whole architecture, a newer version of the SBE template is released. The change is transparent for customers and they can continue to use their current SBE template. Euronext evaluates upfront if a change has an impact on external specifications and therefore if it requires a notification or not.
- Euronext needs to implement hot fixes in production. As Euronext implements incremental changes on the latest supported SBE template version, if a fix requires a change of the SBE structure internally, then a newer version of the SBE template will be released. In any case the change will only impact internal systems and customers will be able to remain on their version of the SBE Template.

In both cases customers will receive from OEG or MDG a newer version of the SBE template than the one they currently use in their applications.