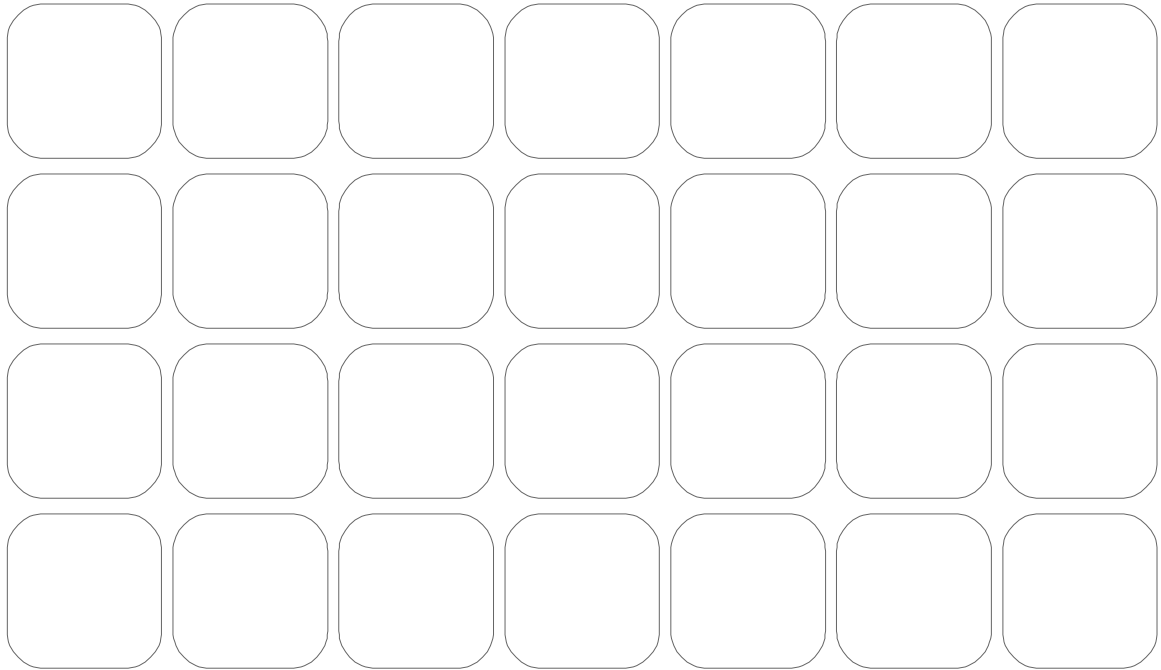


BrokerML
Technical Overview



Tullett Liberty

Version History

Version: 2.3
Version date: 1st May 2003

Acknowledgements to FpML standards team

Synopsis

This document describes the Tullett standard vocabulary for exchanging deal, trade and product information– BrokerML. BrokerML is largely based on FpML 2.0 and 3.0 standards with extensions required to support the broking business.

Document Control

Status: Issue
Authorised By: Peter Meekings
Distribution: As required

Document History

Version:	Date:	Description:
2.1	19 th March 2003	Document created.
2.1.1	24 th March 2003	Revisions and reorganisation.
2.2	27 th March 2003	Released for review.
2.3	1 st May 2003	Changes to the look and feel.

References

Version:	Date:	Document:	Location
1.1	26 th February 2002	BrokerML Design Guidelines v1.1	
1.0	14 th May 2001	FpML 1.0	http://www.fpml.org/spec/2001/rec-fpml-1-0-2001-05-14
2.0	10 th February 2003	FpML 2.0	http://www.fpml.org/spec/2003/rec-fpml-2-0-2003-02-10
1.0	16 th March 2001	FpML Architecture 1.0	http://www.fpml.org/spec/2001/rec-fpml-arch-1-0-2001-03-16

Contents

1	Introduction	5
2	BrokerML Scope.....	6
2.1	BrokerML Business Architecture Framework.....	6
2.2	Supported Business Processes	6
	Product Coverage	7
2.3	BrokerML Guidelines and Standards for XML Design.....	8
	Character encoding and character repertoire	8
	Data Types.....	8
3	BrokerML Business Architecture Overview	10
3.1	Introduction	10
3.2	Component Definitions.....	10
	Interpreting the Diagrams	10
3.3	Business Component Framework	10
3.4	The <i>notification</i> Component	11
3.5	The <i>deal</i> Component	11
3.6	The <i>trades/trade</i> Component	12
3.7	The <i>product</i> Component	12
3.8	Coding schemes.....	13

Figures

Figure 1 – BrokerML Documents	5
Figure 2 - Data Exchange Between Brokers and Traders.....	7
Figure 3 - Graphical Representation of a DTD	10
Figure 4 - Graphical representation of a deal notification	11
Figure 5 - Graphical representation of deal information	11
Figure 6 - Graphical representation of a trade.....	12
Figure 7 - Graphical representation of a product.....	13

Copyright© 2003 by companies in the Tullett Liberty Group.

This document contains information proprietary to companies in the Tullett Liberty Group, and may not be reproduced, disclosed or used in whole or part without the express written permission of the Tullett Liberty Group.

1 Introduction

Tullett Liberty is involved in several initiatives in pursuit of providing practical straight through processing services to its customers and counter parties. BrokerML has been developed to support these services. BrokerML (Broker Mark-up Language) is the title given to the Tullett Liberty development of an XML (Extensible Markup Language) vocabulary for these purposes.

The BrokerML architecture is capable of representing all data items that Tullett Liberty currently capture for any given product. It contains comprehensive information about a given deal and is the standard XML data interchange structure within Tullett Liberty. If required, a BrokerML message can easily be translated into FpML, FIX or other such message type, as BrokerML has more extensive coverage of products than any of these standards. Every effort has been made to comply with and adopt the best practices that have emerged from these standards, most notably the FpML standard.

BrokerML will be extended from time to time as new product requirements dictate.

The following diagram illustrates the structure of the technical documentation describing and supporting the BrokerML standard.

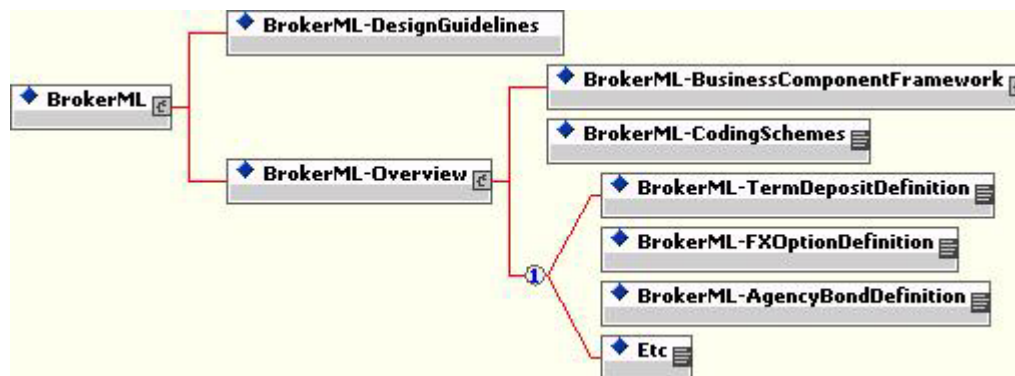


Figure 1 – BrokerML Documents

2 BrokerML Scope

BrokerML defines:

- A Business Architecture Framework that codifies the transaction lifecycle for wholesale financial trading.
- Business specific information about data interchange to enable straight through processing through this lifecycle, the most important being the products that are traded.
- XML Design Guidelines to structure this information.

The following sections introduce these aspects of BrokerML.

2.1 BrokerML Business Architecture Framework

BrokerML has been developed within the guidelines defined in the Tullett Liberty [standards for XML Design](#).

Although naturally closely aligned to the business architecture framework in [FpML 2.0](#) defined by the various Product Working Groups of the FpML organisation, the Tullett Liberty standard differs slightly.

Notable exceptions include:

- BrokerML defines one or more trades within a deal.
- BrokerML has more detailed definition of the organisations involved in a transaction such as agents, executing, trading and billing organisations etc. This facilitates capturing information for trading on global books and flexible invoicing amongst other benefits.
- BrokerML identifies trading and broking personnel as well as support staff involved in a particular transaction. This assists the process of information reconciliation in case there are exceptions.

2.2 Supported Business Processes

The following diagram summarises the business process that may occur in support of basic trading functions. It includes price discovery and order management through to confirmation. It does not include processes associated with the decision support for entering into a financial transaction, such as pricing transactions and interaction with trading books, credit and risk systems etc.

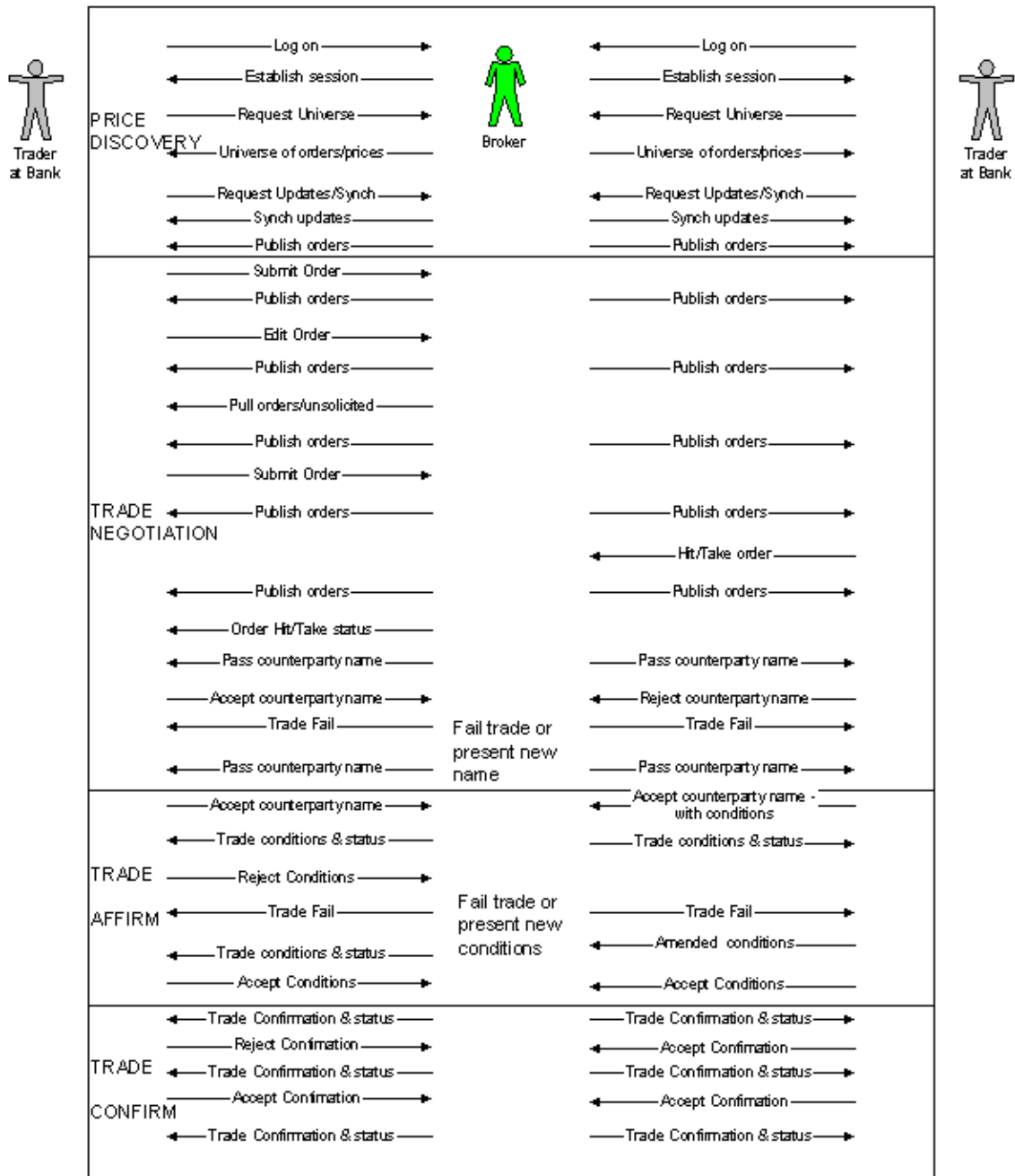


Figure 2 - Data Exchange Between Brokers and Traders

BrokerML is initially restricted to supporting post trade business processes. BrokerML may be extended later to support pre-trade business processes.

Product Coverage

BrokerML supports applications and services that provide proof of principle services in support of post trade affirmation and confirmation of the following products:

- Spot FX & Outright FX
- Forward FX
- Foreign Exchange Options
- Cash Bonds
- Spread (Bonds)
- Basis (Bond against Future)
- Forward Rate Agreement
- Single currency Interest Rate Swap (fixed vs Float)

- Basis Swap
- Asset Swap (single currency IRS against Cash Bond)
- Interest Rate Cap
- Interest Rate Floor
- Interest Rate Swaption (European, Bermudan and American Styles; Cash and Physical Settlement)
- Extendible and Cancellable Interest Rate Swap Provisions
- Mandatory and Optional Early Termination Provisions for Interest Rate Swaps
- FX Resettable Cross-Currency Swap

2.3 BrokerML Guidelines and Standards for XML Design

The current version of BrokerML (2.0) is DTD based. It has been developed using the [BrokerML Design Guidelines 1.1](#). It is closely aligned to the XML design guidelines in [FpML Architecture Version 1.0](#) defined by the FpML Architecture Working Group, the Tullett Liberty XML design guidelines differ very slightly from them.

The most notable exception is:

- Containers for arrays - Within BrokerML arrays are found within containers, regardless of whether the container has any attributes of its own.

BrokerML 2.0 adopts the same approach to DTD structures as the FpML standard – all asset classes supported by BrokerML are defined within a single DTD. This approach simplifies the task of maintaining consistent versions of XML components.

The initial and primary purpose of BrokerML is to support and facilitate information exchanges for trade processing. Therefore the underlying technologies for its implementation should be very stable in order for them not to hinder this primary purpose. The most stable XML standard for such implementations still remains to be based on DTD technologies. However, it is clearly sensible to move to XML Schema standards as soon as it is practical, because of their numerous advantages and potential to enable more precise and consistent levels of information exchange. Probably the main consideration is the internal referencing solution preferred by the FpML standard as it moves to XML Schemas standards. The BrokerML team will track, and is likely to incorporate, any recommendations made by the FpML Architecture Working Group for migrating to XML Schema Definition language and other related standards.

Character encoding and character repertoire

2.3.1 Character Encoding

Producers of BrokerML documents must encode such documents using either UTF-8 or UTF-16. Consumers of BrokerML documents must be able to process documents encoded using UTF-8, as well as documents encoded using UTF-16. For more information, see <http://www.w3.org/TR/REC-xml#charencoding>.

2.3.2 Character Repertoire

BrokerML element content, as well as values of the BrokerML `id` and `href` attributes, may use any valid XML characters. For more information, see <http://www.w3.org/TR/REC-xml#charsets>.

Data Types

BrokerML 2.0 uses a subset of the built-in datatypes (both primitive and derived datatypes) as defined in XML Schema Part 2: Datatypes, W3C Recommendation 02 May 2001. The built-in datatypes are described at:

<http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/ - built-in-datatypes>

The built-in datatypes used in BrokerML are the following:

- `boolean`
- `date`
- `decimal`
- `integer`

- `nonNegativeInteger`
- `positiveInteger`
- `string`
- `time`.

The set of valid literals for each data type are those defined in the XML Schema specification as being its lexical space. Additional constraints are imposed by BrokerML on the `date` and `time` built-in data types as described below.

2.3.3 *date*

All elements of type `date` in BrokerML must contain date values with the format CCYY-MM-DD where “CC” represents the century, “YY” the year, “MM” the month and “DD” the day. The CCYY field must have at least four digits, the MM and DD fields exactly two digits each; leading zeroes must be used if the field would otherwise have too few digits. A following time zone qualifier is not allowed and year values must be in the range 0001 to 9999. For example, 25 May 2000 would be represented in BrokerML as 2000-05-25. All dates and times in BrokerML are used alongside a business centre or a time zone.

2.3.4 *time*

All elements of type `time` in BrokerML must represent daily recurring instants of time values with the format hh:mm:ss where “hh”, “mm” and “ss” represent hour, minute and second respectively. The hh, mm and ss fields must have exactly two digits each; leading zeroes must be used if the field would otherwise have too few digits. BrokerML imposes the further restriction that the second (ss) field must be ‘00’ and a time zero qualifier is not allowed. For example, 00:00:00 (midnight), 01:00:00 (1:00am), 12:00:00 (midday), 23:30:00 (11:30pm). All dates and times in BrokerML are used alongside a business centre or a time zone.

3 BrokerML Business Architecture Overview

3.1 Introduction

BrokerML uses the capabilities of the XML standard to build a well-structured definition of the information exchanged in support of the various wholesale broking business processes. Definitions of information in the form of XML elements are logically grouped together within XML entities based containers and they together form the basic components of BrokerML.

3.2 Component Definitions

Interpreting the Diagrams

The DTD source shown below is graphically represented in Figure 2. Important features of the diagram are highlighted, which include:

- Graphical representation of an XML entity definition
- Sequence indicators, i.e. comma (,) and vertical bar (|)
- Content specifications, i.e. text or sub-elements
- Occurrence indicators, i.e. can appear zero or once (?), can appear one or more times (+), can appear zero or more times (*).

```
<!ENTITY % BML_Root "SubElementA?,SubElementB+ ">
<!ELEMENT SubElementA (LeafElementA*,(LeafElementB | LeafElementC))>
<!ELEMENT SubElementB
(LeafElementA,LeafElementB,LeafElementC,LeafElementD) *>
<!ELEMENT LeafElementA (#PCDATA)>
<!ELEMENT LeafElementB (#PCDATA)>
<!ELEMENT LeafElementC (#PCDATA)>
<!ELEMENT LeafElementD (#PCDATA)>
```

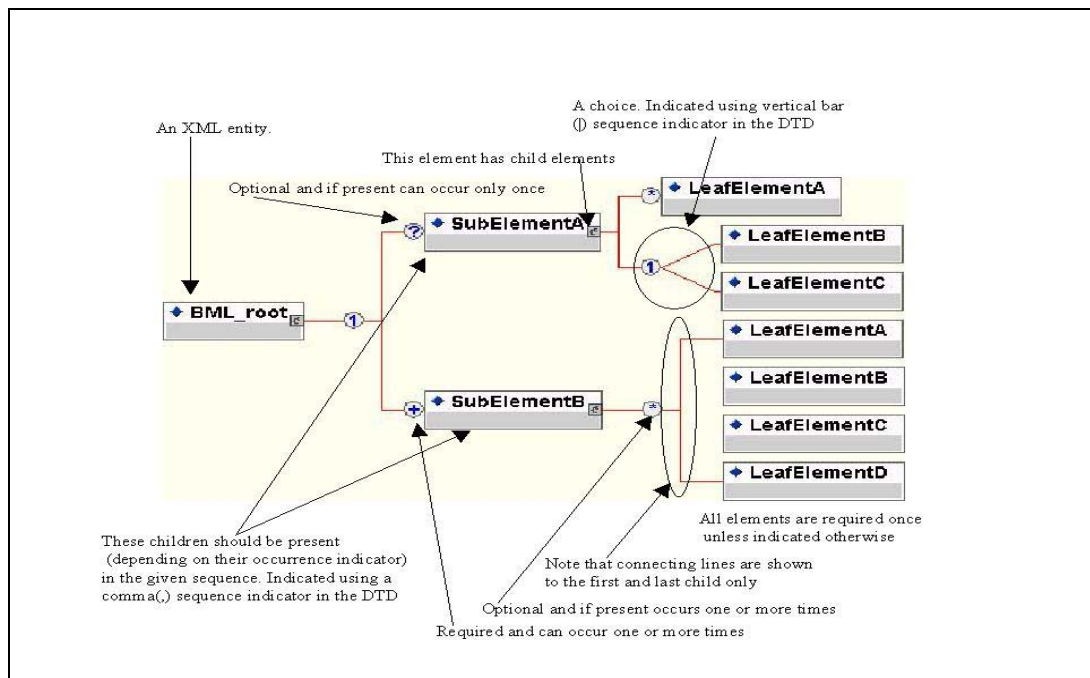


Figure 3 - Graphical Representation of a DTD

3.3 Business Component Framework

The main business components in the BrokerML business component framework are

- Notification – This component contains information about notification of a deal that has been brokered by the Tullett group.

- Deal - This component contains economic and other information about a deal brokered by TullettLiberty. A deal is a single transaction between a trader and a Tullett Liberty broker, and may comprise a number of separate trades. Complex strategies such as spread and basis trading, also asset swaps, are represented by a deal component with many trade components.
- Trades/Trade - This component contains economic and other information about individual trades within a deal brokered by Tullett Liberty.
- Product - This component contains economic and other information about individual products that have been traded within a deal brokered by the Tullett group.

3.4 The *notification* Component

The *notification* is the top-level component for notification of a transaction in BrokerML. This notification has details about a deal that has been done between two or more parties.

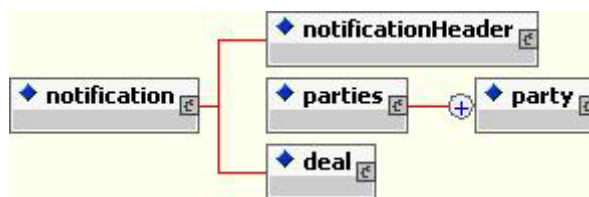


Figure 4 - Graphical representation of a deal notification

A BrokerML *notification* contains three components:

- General details about the notification – *notificationHeader*.
- Details about two or more parties involved in the notification – *parties/party*.
- Details about the various components of transactions that constitute a deal between two or more parties – *deal*.

3.5 The *deal* Component

The *deal* component is the top level BML element containing one or more trades. It represents the financial contract between two or more parties. The *deal* element contains all the information required to confirm this contract.

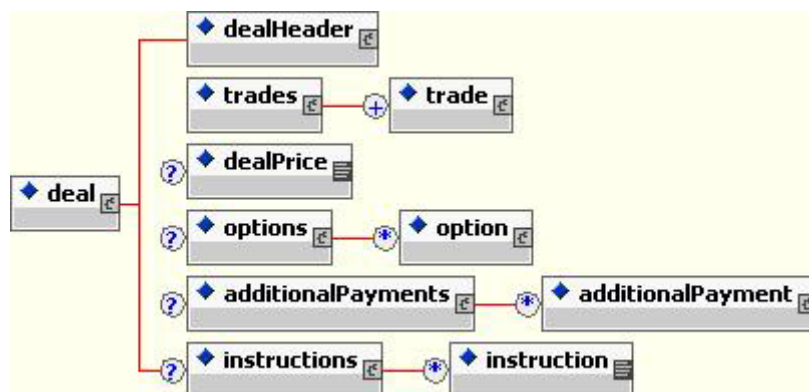


Figure 5 - Graphical representation of deal information

It contains

- A deal header – *dealHeader*.
- One or more trade details – *trades/trade*.

- The optional price of the deal – *dealPrice*.
- Details of zero or more optional conditions on the completion of the deal – *options/option*.
- Optional details of any payments additional to the principal economic details of the deal (e.g. brokerage fee, regulatory payments etc.) – *additionalPayments*.
- An optional set of notes to settlement staff – *instructions*.

3.6 The *trades/trade* Component

A trade is an agreement between two parties to enter into a financial contract for exchanging products in return for a payment. The *trades/trade* component in BML contains the economic information necessary to execute individual transactions that are part of the overall transaction contained in a *deal* component.

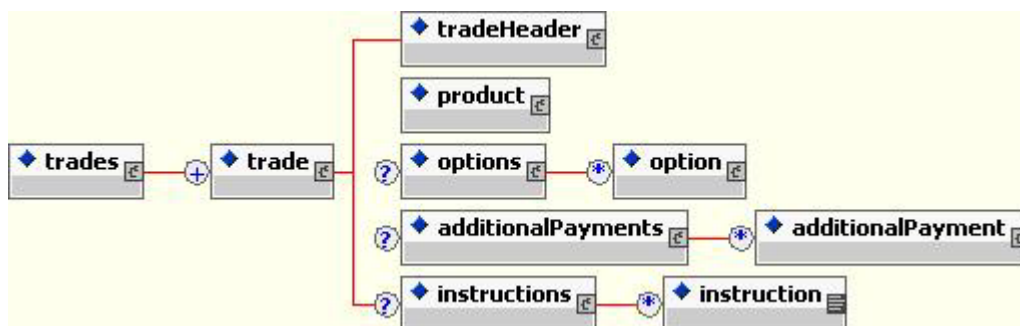


Figure 6 - Graphical representation of a trade

It contains

- Information common to all the components of an individual financial transaction – *tradeHeader*.
- A description of the product traded, where a ‘product’ is an abstract concept and is a place holders for actual financial products such as bonds, swaps, foreign exchange etc. – *product*.
- Details of zero or more optional conditions on the completion of the trade – *options/option*.
- Optional details of any payments additional to the principal economic details of the trade (e.g. brokerage fee, regulatory payments etc.) – *additionalPayments*.
- An optional set of notes to settlement staff – *instructions*.

3.7 The *product* Component

The *product* component specifies the financial instrument being traded. This component captures the economic details of the trade. Because of the complexity of the OTC product domain such as Interest Rate Derivatives that BML covers, composing these products from various building blocks is a key aspect of the design approach.

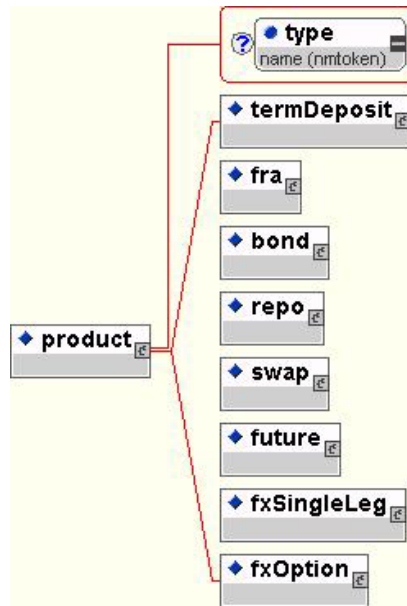


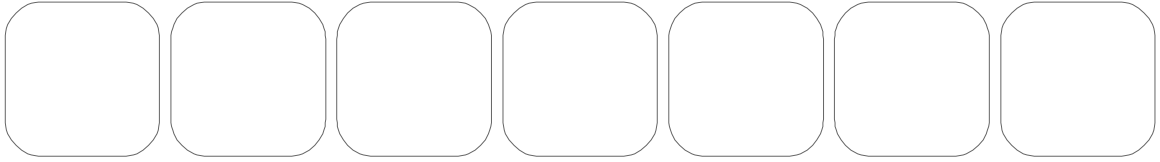
Figure 7 - Graphical representation of a product

The individual product definitions are defined in individual specifications for these products.

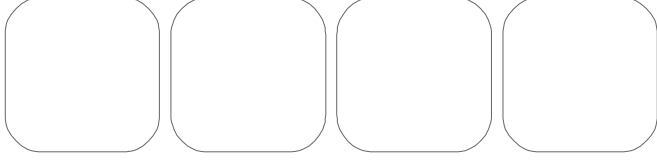
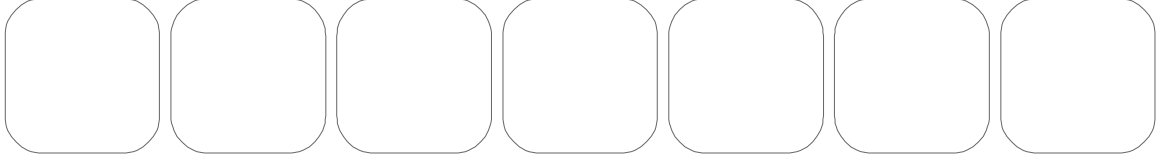
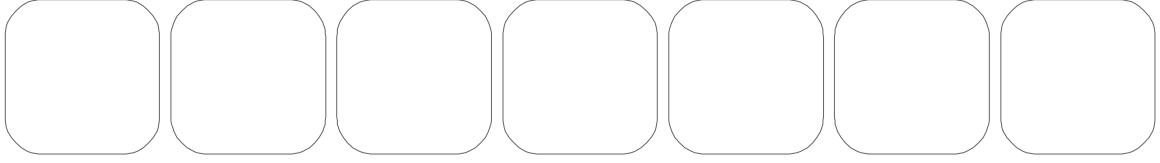
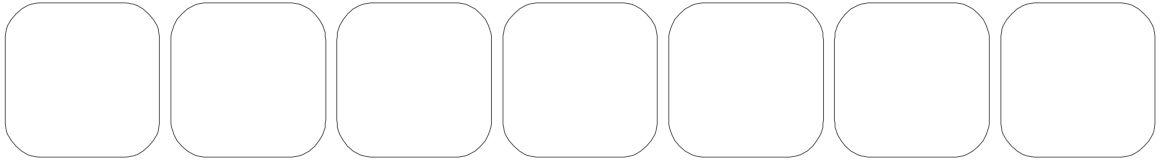
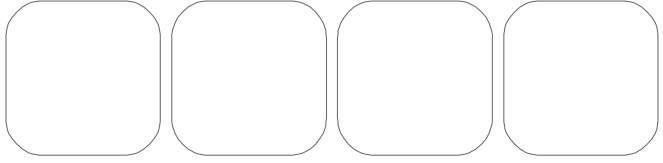
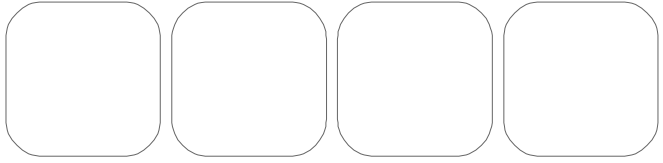
3.8 Coding schemes

A necessary feature of a portable data standard is both an agreed set of elements and an agreed set of permissible values (the value domain) for those elements, called Reference data. A BrokerML document exchanged between two parties is only understandable when both of the parties use common coding schemes to populate elements.

For these reasons, BrokerML uses standard XML Schemes to identify the permitted values for an element. In each case, the reference Scheme is identified by a URI (Uniform Resource Identifier). The URI either identifies a well-known external standard such as ISO 4217, or where no well-established standard exists, a Tullett Liberty standard.



Tullett Liberty
Cable House,
54-62 New Broad Street,
London,
EC2M 1ST
ENGLAND
www.tullib.com



Tullett Liberty