Draft Technical Note:

FpML Validation Language Requirements

Abstract:

This document sets out the requirements for a validation constraint language for FpML. This language will be used to specify constraints on FpML documents that cannot be expressed in DTD or Schema form, and that will define what constitutes a meaningful trade. The lifetime of this document is limited – its sole purpose is to assist the Validation Working Group in gathering initial feedback and to help it to prioritize requirements and select a constraint language.

This version:

http://www.fpml.org/spec/ValidationRequirements2003-03-04

Document expires: 25th March 2003

Status of this Document:

The Validation Working Group is issuing this document in order to gather feedback from product working groups and the community at large on the requirements for the FpML validation constraint language. As a result of this feedback the working group hopes to be able to prioritize the requirements, and to put in place evaluation criteria.

This document expires on March 25, 2003. Feedback should be submitted before then via the FpML issues form at http://www.fpml.org/mailing-lists/fpmlmail/issues.asp or by e-mail to val-wg-chair@fpml.org.

Editor

Christian Nentwich - Systemwire

Authors

Philip Allen - Decisionsoft

Richard Carter - JP Morgan

Daniel Dui - University College London

Wolfgang Emmerich - Zuhlke Engineering

Steven Lord - UBS Warburg

Brian Lynn - Gem Soup

Gareth Reakes - Decisionsoft

Robert Stowsky - Brook Path Partners

1. Introduction

By checking an FpML trade against a Schema or DTD we can determine whether its syntactic structure is valid. Unfortunately, syntactic validity cannot guarantee that a trade will be in any way meaningful, and FpML does not presently cover the needs for further validation. This includes syntactic validation that cannot be performed using XML Schemas or DTDs, for example co-constraints, more rigorous checks on the different product types to ensure that they are meaningful, and business-level validation that may be specific to a particular institution.

This document sets out the requirements for a validation language for FpML. This validation language has to fulfil two roles: a normative role in constraining FpML and defining what constitutes a meaningful trade; and an operational role in that constraints should be executable, or amenable to translation into an executable format. It is envisaged that a set of constraints will be published for each particular product type, for inclusion with each version of FpML. Institutions should then be able to specify which of these constraints they want in place on trades they are prepared to receive, and to give guarantees about the properties of trades that they are generating.

In the following we set out the high-level goals for the validation language, followed by a break-down of the requirements necessary to achieve the goals. The requirements are further divided into functional and non-functional sections, to distinguish requirements on features from requirements on intangible properties such as usability. The final section gives some more details on what kind of feedback we are seeking.

2. Goals

Goal 1. FpML needs to supply validation rules with each version of FpML for community wide issues. The validation constraint language should facilitate this.

Goal 2. The validation language has an operational role. Institutions should be able to execute validation constraints written in the constraint language – this includes constraints supplied with FpML and internal constraints.

Goal 3. Validation should focus on semantic or business validation. It is assumed that XML parsers are used for XML syntax validation based on the FpML DTD/Schema (for both well-formedness checking and syntactic validation).

Goal 4. It must be possible to provide GUI tools to enable business analysts of institutions to formulate and update constraints.

3. Requirements

1	Ability to express different classes of constraints. The constraint language must be expressive enough to handle a variety of constraint types.
1.1	There should be convenient mechanisms for expressing the following two-element co-constraints: existence-existence, value-existence, existence-value and value-value. *Note: This reflects the observation that these types of constraints occur very frequently.*
1.2	There must be a convenient mechanism for expressing foreign key reference constraints.

1.3	There must be a convenient mechanism for expressing uniqueness constraints.
1.4	The language must support FpML in the transition to XML Schema, by supporting constraint types that cannot be expressed in DTDs, even though they may be supported by XML Schema. *(See also: 1.2, 1.3)*

2	It must be possible to validate trades against external data sources. *(See Goal 2)*
2.1	The constraint language must allow seamless referencing of external static data.
2.1.1	The constraint language should be able to handle external data that is not available in XML form.
2.1.2	The language should retain some degree of location transparency. *Notes: This is to avoid people having to rewrite their constraints when their data sources move.*
2.2	The constraint language must provide a call-out mechanism for referencing dynamic data or data provided by legacy systems.

3	It must be possible to translate the constraints expressed in the language into an executable format, or to provide an interpreter. *(See Goal 2)*
3.1	The formalism chosen must be amenable to automated translation into an internal representation of an execution engine.
3.2	The translated constraints must be executed efficiently, so that between 100 and 300 constraints can be evaluated in a reasonable amount of time. *Notes: These figures will need to be adjusted once we understand the scope of the task better.*
3.3	It would be preferable to choose an XML encoding for the constraint language as this would ease tool and compiler construction, and enable direct manipulation through off-the-shelf editors.

4	FpML builds on a number of data types, for example dates. It must be possible to manipulate these data types efficiently. *Note: This can be achieved through native data type support or a scripting mechanism.*

5	There needs to be a way of associating reporting data with constraints, to provide diagnostic feedback in case of constraint violations.
5.1	Reports need to be presented in XML, to enable integration with the FpML messaging architecture.

6	Many FpML elements may appear in multiple locations. The language must facilitate the specification of constraints regardless of element location, to enable reuse.

4. Non-functional Requirements

7	The language should be easily comprehensible and therefore use concepts that members of the FpML community should be familiar with.
7.1	The constructs in the language should have a well-defined meaning, preferably a formally defined semantics.
7.2	The language shall provide convenient mechanisms for expressing common types of constraints. *(See 1.1)*
7.3	It must be possible to translate the language from its XML encoding into a format that is suitable for presentation in specification documents.

8	The constraint language should be in a declarative format for reasons of conciseness and ease of use. *Notes: This does not mean that no imperative mechanisms should be available, for example for performing complex operations on atomic data types such as dates. This is also in no way a constraint on how the language is implemented.*

9	The validation language should make use of existing W3C XML recommendations.

5. Feedback

The Validation Working Group is currently evaluating a number of candidate constraint languages against this requirements specification. In order to help us prioritize requirements, it is important that we obtain feedback on what the community feels are the most important issues in this area.

We are happy to receive any kind of feedback on this document. In addition, you may want to consider these questions:

- What should be the scope of the constraint language? Should the language be supported by implementers across the industry? Or should the scope be narrowed, so that the language is used to express constraints for the specifications, perhaps including a reference implementation, but leaving implementation entirely open?
- There is no standardised constraint language that meets all of the requirements. What is the single most important requirement, functional or otherwise, that you would expect such a language to fulfil?
- Are you working for a company/institution that could use the constraint language as an operational mechanism? If so, do you have any further functional requirements? What about interfacing requirements? What level of technical expertise would you expect from a constraint writer?

Please send feedback using the FpML issues form at http://www.fpml.org/mailing-lists/fpmlmail/issues.asp or by e-mail to val-wg-chair@fpml.org.