Changes in FpML 5.11-Recommendation

compared to FpML 5.10 published February 12, 2018

Syndicated Loan (Confirmation View) (changes in 5.11 compared to 5.10)

Major design approach changes have resulted in significant revision to the Loan FpML schema for version 5.11. These changes better support uniformity of design, extensibility, and simplification (e.g. reducing the number of acknowledgement, exception, and retracted messages within the schema). Changes include:

1. *'Trade' vs. 'Transfer' Concepts*
    - The differentiation between the term 'trade' and 'transfer' was eliminated from the Loan FpML schema as this was creating confusion. As such:
        - The 'trade' ('LoanTrade' complex type) structure has been remodeled such that the elements of 'price', 'transferFee', the models 'LoanTradingCounterpartyCashSettlementRules.model' and 'LoanTradingParticipationSettlementTerms.model' have all been placed on an optional sequence, to be included when the communication of the loan trade event is between counterparties (as opposed to between counterparty and agent).
        - The 'loanAllocationSettlement' ('LoanAllocationSettlementEvent' complex type) event has been remodeled such that the element of 'fundingFactors' has been placed on an optional sequence, to be included when the expression of the settlement of the allocation is between counterparties (as opposed to between counterparty and agent).
    - These changes place more onus on the party implementing the standard to understand when it is appropriate to send the optional sequences mentioned above. XML examples describing the way to appropriately use these new optional sequences have been produced.

2. *Abstract Type Structural Changes*
   Added or changed existing abstract types to support better organization within the above-mentioned substitution groups.
    - 'AbstractFacilityRateChangeEvent'
        - This abstract type was added to communicate rate change events at the facility level.
    - 'AbstractLoanContractPaymentEvent'
        - This abstract type allows nonrecurring fee payment events at the loan contract-level to derive from a single structure.

- ‘AbstractLoanAllocationSummary’
    - Corrected the type for the attribute ‘id’ to xsd:ID (previously missing a type).
    - Changed ‘amount’ element to NonNegativeMoney type.
- ‘AbstractLoanTradeSummary’
    - Corrected the type for the attribute ‘id’ to xsd:ID (previously missing a type).
- ‘AbstractLoanEvent’
    - Changed the inheritance so that the ‘AbstractLoanEvent’ no longer inherits AbstractEventRequireId, but rather just includes an ‘eventIdentifier’ (BusinessEventIdentifier complex type).
    - An optional ‘previousEventIdentifier’ element was added to this structure to reference an event that came chronologically before the event being conveyed by the message.
- ‘AbstractLoanAllocationPaymentNotification’
    - The ‘payment’ element (LoanAllocationPayment complex type) should be used to express a rollup of all cash payable amounts related to allocations (so as to mirror the use of the comparable ‘eventPayment’ element in the AbstractServicingNotification), otherwise the element is redundant with the ‘cashPayable’ element within the allocation events themselves. This abstract type was retired from the 5.11 schema, and the ‘payment’ element (LoanTradingPayment complex type) was added to AbstractLoanTradingNotification, with 0..n cardinality.
- ‘AbstractLoanAllocationSummary’
    - This abstract type was removed, as it was an additional layer that created confusion with multiple ‘id’ attributes. The removal of this abstract type fosters consistency throughout the schema.
- ‘AbstractLoanPartyProfileNotification’
    - The AbstractLoanPartyProfileStatement in 5.11 is not backward compatible due to the name change of this abstract type, as described in the previous section.
- ‘AbstractLoanTradePaymentNotification’
    - This abstract type was retired for the same reason as the AbstractLoanAllocationPaymentNotification, described above.
- ‘AbstractLoanTradeSummary’
    - This abstract type was removed, as it was an additional layer that created confusion with multiple ‘id’ attributes. The removal of this abstract type fosters consistency throughout the schema.
- AbstractLoanTradingNotification’
    - This abstract type was split into ‘AbstractLoanTradeNotification’ and ‘AbstractLoanAllocationNotification’ in order to provide the ability for structures that inherit these abstract types to accurately reference supporting structures in each. The original abstract type contained only an ‘allocationReference’ element. A ‘tradeReference’ element was created and included in ‘AbstractLoanTradeNotification’.

- IdentifiedAssetWithParty
    - Extended Asset to include the 'PartyAndAccountReferences.model', description, and 'instrumentId'.
    - This looks similar to the shared structure 'IdentifiedAsset', but with the above-mentioned included model. We decided not to inherit directly from 'IdentifiedAsset', as due to the inheritance hierarchy adding the model made the relationship of the party/account reference to the 'instrumentId' element unclear.
    - Changed all asset identifiers (Deal, Facility; Loan Contract and Letter of Credit already inherited 'ContractIdentifier') to inherit the new 'IdentifiedAssetWithParty' structure.
- NonRecurringFeePayment
    - This abstract type was misnamed without the 'Abstract' prefix. This was renamed as 'AbstractNonRecurringFeePayment' in 5.11 and is not backward compatible.

3. *Inheritance Structural Changes*
   Several events have been redesigned to inherit consistently from the same abstract type as other events occurring at the same structural level (contract level, facility level, LC level, trade level, allocation level). The following events are impacted:
    - amendmentFeePayment
    - breakageFeePayment
    - facilityExtensionFeePayment
    - fundingFeePayment
    - miscFeePayment
    - upfrontFeePayment
    - waiverFeePayment

4. *Substitution Group Structural Changes*
    - Substitution groups utilized by the loan servicing type messages were not organized in consistent ways. I.e. some substitution groups seemed to aggregate events by fee-types in one instance, while others seemed to aggregate events by structural location. For example, in FpML 5.10 the 'facilityFeePaymentGroup' is utilized for non-recurring fee events at *both* the facility and contract level.
    - To remedy the inconsistent application of substitution groups across the loan servicing type message set, the substitution groups were redesigned and reorganized into:
        - 'facilityEventGroup' – this group now contains all facility-level events based on the 'AbstractFacilityEvent' type.
        - 'loanContractEventGroup' – this group now contains all loan contract-level events based on the 'AbstractLoanServicingEvent' type.
        - 'lcEventGroup' – this group now contains all letter of credit-level events based on the 'AbstractLcEvent' type.
        - 'loanTradeEventGroup' – this group now contains all trade-level events based on the 'AbstractLoanTradeEvent' type.

        - 'loanAllocationEventGroup' – this group now contains all allocation-level events based on the 'AbstractLoanAllocationEvent' type.

5. *'Event' Structural Changes*
   Events have been added or redesigned to be more consistent across the Loan FpML schema. Changes include:
   - 'AccrualOptionChangeEvent'
     - There was no 'AccrualOptionChangeEvent', even though other similar events were expressed as such (e.g. 'AccruingFeeChange'). The elements related to this business event were described as 'loose' elements within a message element wrapper that was derived directly from 'AbstractContract Notification'. As such, a new 'AccrualOptionChangeEvent' was created that inherits from 'AbstractFacilityEvent' and extends it with the necessary elements. This also allows this event to reside within the 'facilityEventGroup' substitution group.
   - 'Prepayment' and 'Rollover'
     - Embedded events have been stripped out of these two structures. Embedded events already existed as stand-alone events, and the approach of embedding events created redundant expression of certain elements, and ambiguity around utilization.
     - 'maturingContracts' – this has been renamed to 'currentContracts'
     - 'currentContracts' – this has been renamed to 'newContracts'
   - Embedded Event Processing
     Going forward, conveying a relationship between events which were previously embedded within other events can be accomplished by one of two ways:
     - Send all events as separate servicing event notifications, relating the notifications via the 'childEventIdentifier'/'parentEventIdentifier' relationship; or,
     - Send all related events within the same 'loanServicingNotification' message element wrapper, relating events via the 'childEventIdentifier'/'parent EventIdentifier' relationship mentioned above.

     XML examples have been produced to explicitly depict this schema change.

6. *Discrete Element Naming Changes*
   - The term 'notification' should be reserved for the purposes of expressing a business event. As such, 'LoanPartyProfileNotification' has been renamed. The word 'notification' has been replaced by 'statement.' This has occurred at both the complex type and element levels.
   - The 'adjustment' event within the 'loanContractEventGroup' substitution group has been renamed to 'loanContractAdjustment' to clarify the intent of the event.
   - The 'type' element within the 'LoanTrade' complex type has been changed to 'marketType' to be more explicit as to the 'type' being communicated and align it with common business terminology. The complex type has not been altered and still derives from the 'LoanTradingTypeEnum' enumeration.

7. *'Notification' Wrapper Structural Changes*

- By reorganizing substitution groups as noted above, it became possible to streamline message element 'wrappers' into one 'LoanServicingNotification' core type. The 'LoanServicingNotification' now includes all loan servicing-level events with 1-to-many cardinality, encompassing what was previously accomplished in the separate notifications. The following notifications are now expressed by the 'LoanServicingNotification' and no longer exist as stand-alone notification structures within the Loan FpML schema:
    - 'loanContractNotification'
    - 'lcNotification'
    - 'facilityNotification'
    - 'loanBulkServicingNotification' – this message element wrapper is no longer necessary, as multiple events can be sent using 1 'loanServicingNotification.'
- The following structures are not included in the newly created 'loanServicingNotification' but have changed since the implementation of FpML v5.10:
    - 'loanAllocationNotification' – this message element wrapper now contains a substitution group with all trade allocation-level events. It also contains the 'settlementTask' element ('LoanAllocationSettlementTask' complex type) for allocation-level tasks, on a choice sequence with the 'loanAllocationEvent Group' substitution group, so that this single message can now be conveyed to represent both allocation events and allocation settlement tasks. Event elements related to the following notifications are now expressed by the substitution group within the 'loanAllocationNotification', and no longer exist as stand-alone notification structures within the Loan FpML schema:
        - loanAllocationConfirmationNotification
        - loanAllocationSettlementNotification
        - loanAllocationSettlementTaskNotification
        - loanAllocationSettlementDateAvailabilityNotification
        - loanAllocationSettlementDateFinalizationNotification
        - loanAllocationTransferFeeDueNotification
        - loanAllocationTransferFeeOwedNotification
    - 'loanTradeNotification' – this message element wrapper now contains a substitution group with all trade-level events. It also contains the 'settlementTask' element ('LoanTradeSettlementTask' complex type) for trade-level tasks, on a choice sequence with the 'loanTradeEventGroup' substitution group, so that this single message can now be conveyed to represent both trade events and trade settlement tasks. Event elements related to the following notifications are now expressed by the substitution group within the 'loanTradeNotification', and no longer exist as stand-alone notification structures within the Loan FpML schema:
        - loanTradeConfirmationNotification
        - loanTradeSettlementTaskNotification
        - loanTradeTransferFeeDueNotification
        - loanTradeTransferFeeOwedNotification
        - loanTransferNotification

        - loanTransferSettlementNotification

8. *Other Complex Type Changes*
   Changes were made to complex type structures or inheritance in order to support uniformity and design consistency. These changes include:
   - 'AccrualOptionChangeEvent'
     - The following complex types were incorporating entire event structures, which was inappropriate. We changed the inheritance for the following to inherit the imbedded complex type only and not the entire event structure:
       - fixedRateOptionChange – also changed name to 'fixedRateOption'
         a. Added 'loanContractReference' to sequence with the complex type
         b. Removed the contract complex type object - there should be a reference to the footer.
       - floatingRateOptionChange – also changed name to 'floatingRateOption'
         a. Added 'loanContractReference' to sequence with the complex type
         b. Removed the contract complex type object - there should be a reference to the footer.
       - accuringPikOptionChange - also changed name to 'accruingPikOption'
       - lcOptionChange – also changed name to 'lcOption'
         a. Added 'letterOfCreditReference' to sequence with the complex type
         b. Removed 'letterOfCredit' complex type object - there should be a reference to the structure in the footer.
   - 'ApplicableAssets'
     - Added 'LoanAllAssetsEnum' to a choice block to all for the explicit declaration of the applicability of all assets.
   - 'ApplicableTransactions'
     - Added 'LoanAllTransactionsEnum' to a choice block to all for the explicit declaration of the applicability of all assets.
   - 'ApplicableCommunicationDetails'
     - 'ApplicableAssets' minimum occurrences allowed changed from 0 to 1.
     - 'ApplicableTransactions' minimum occurrences allowed changed from 0 to 1.
   - 'CashPayable'
     - Replaced 'PayerReceiver.model' with 'SimplePayerReceiver.model' to eliminate the account references that are not used.
   - 'DealIdentifier'
     - Created a 'dealIdentifier' element along with href pointers, that was added to all relevant statements and notifications.
   - 'DealSummary'
     - Created a 'dealSummary' element along with href pointers, that was added to all relevant statements and notifications.
   - 'EventPayment'

- Optional 'SettlementInstructions' complex type included in the structure to meet a specific business use case expressed by the working group.

- 'FacilityPrepayment'
  - Renamed 'Prepayment' to 'FacilityPrepayment' and removed 'prepaymentFee' from this event. This resolves an issue with similar naming of elements across different asset classes.
- 'FacilityPrepaymentFeePayment'
  - Created a 'FacilityPrepaymentFeePayment' element to capture the event of paying the prepayment fee. This event corresponds to the 'Facility Prepayment' event. This change also resolves an issue with similar naming of elements across different asset classes.
- 'EventPayment'
  - Optional 'SettlementInstructions' included in the structure to meet specific business use cases.
  - Inheritance changed to 'LoanSimplePayment' (which inherits 'Simple Payment') to remove elements that are not used by the syndicated loan market.
- 'LoanAllocation'
  - Replaced 'BuyerSeller.model' with 'SimpleBuyerSeller.model' to facilitate the syndicated loan asset class.
- 'LoanAllocationNotification'
  - Added LetterOfCreditDetails.model and LoanContractDetails.model on an optional choice block (0..n) to allow reference structure optionality.
- 'LetterOfCredit'
  - Added a missing 'id' attribute to facilitate reference by other objects
- 'LetterOfCreditSummary'
  - Added a missing 'id' attribute to facilitate reference by other objects.
  - Changed structure to inherit 'FacilityContractIdentifier' for consistency of schema design.
- 'LoanAllocationSettlementEvent'
  - Changed the 'amount' element to 'NonNegativeMoney' type and added an optional 'SimplePayerReceiver.model' to clarify cash flow for the following elements within the 'fundingFactors' element:
    - 'delayedCompensation'
    - 'costOfCarry'
    - 'economicBenefits'

    **The actual accrual amounts within the structure remain 'MoneyWithParticipantShare' types in order to account for scenarios such as inverse accrual rates and unfunded facilities.
- 'LoanAllocationSummary'
  - Created this concrete type for consistency of schema design, due to the removal of the AbstractLoanAllocationSummary abstract type.
- 'LoanContract'
  - Added a missing 'id' attribute to facilitate reference by other objects.
- 'LoanContractAdjustment'

- Changed the element name from 'adjustment' to 'loanContractAdjustment' for clarity.

- 'LoanContractNotification'
  - LoanContractDetails.model changed from 1 to 1..n to support the idea of multiple existing contracts rolling into multiple new contracts (i.e. one to many, many to one, many to many) within the Rollover event.
- 'LoanContractSummary'
  - Added a missing 'id' attribute to facilitate reference by other objects.
  - Changed structure to inherit 'FacilityContractIdentifier' for consistency of schema design.
- 'LoanSimplePayment'
  - Changed 'paymentDate' to inherit 'AdjustableDate' instead of 'RelativeDate,' as 'RelativeDate' is not a concept that applies to a loan payment.
- 'LoanTrade'
  - Changed 'type' element to 'marketType' to clarify between primary or secondary.
- 'LoanTradingDelayedCompensation'
  - Changed 'amount' element to type 'Money' in order to account for potential all-in negative values.
- 'LoanTradingPayment'
  - Optional 'SettlementInstructions' object included in the structure to meet specific business use cases.
  - Inheritance changed to LoanSimplePayment (which inherits SimplePayment) to remove elements that are not used by the syndicated loan market.
- 'LoanTradingNonRecurringFee'
  - Significant changes were made to this structure, including a more exhaustive representation of potential identification and reference of non-accruing fees to asset structure, and more optionality to include miscellaneous fees within this structure. This complex type was also renamed to 'LoanTradingNonAccruing Fee' and removed from the 'LoanTradingSettlementAccruals.model' structure. For more detail see the section on 'LoanTradingSettlementAccruals.model' changes below.
- 'LoanTradingSettlementTaskDates'
  - Changed the name of the enum to 'LoanTaskDates' to make it more universally usable (as opposed to specific to the loan trading settlement scenario). Values within the enum were NOT changed.
- 'ParentEventIdentifier'
  - Created an optional structure which inherits the 'BusinessEventIdentifier' structure and extends by adding a required 2..n element 'childEventIdentifier' (based on 'BusinessEventIdentifier') to describe all the underlying child events. This provides the ability to reconcile when related event notifications may be missing, or events within the same 'LoanServicingNotification' structure may be missing (a business validation point).

- ‘ParentTaskIdentifier’
    - Created an optional structure which inherits the ‘BusinessTaskIdentifier’ structure and extends by adding a required 2..n element ‘childTaskIdentifier’ (based on ‘BusinessTaskIdentifier’) to describe all the underlying child tasks. This provides the ability to reconcile when related task notifications may be missing, or events within the same ‘LoanServicingNotification’ structure may be missing (a business validation point).

9. *Changes to ‘Statements’*
   To support uniformity and design consistency the following changes were made to ‘statement’ types:
    - ‘DealStatement’
        - Added a sequence with 1..n optionality ahead of the ‘facilityGroup’ substitution group. Originally the head of the substitution group was set to 1..n which wouldn’t accomplish the outcome we want: multiple facilities of potentially multiple (or same) types. By adding the sequence and changing that sequence to 1..n we are able to include multiple facilities of either the same or different type.
    - ‘FacilityOutstandingsPositionStatement’
        - This statement was removed in favor of adding an optional outstandings structure, using the ‘facilityPosition’ element, to the ‘FacilityPositionStatement’.
    - ‘LoanPartyEventInstructionOverrideStatement’
        - This statement was removed since optional settlement instruction elements now exist in all servicing and trading notifications. Related structures (i.e. ‘overrideId’) were removed.
    - ‘LoanPartyProfileStatement’
        - Within this statement, the ‘communicationDetails’ element contained ‘purpose,’ ‘personReference,’ and ‘businessUnitReference.’ These have been replaced by a structure that contains ‘relatedPerson’ and ‘relatedBusinessUnit’ which are both elements inherited from the broader schema. The ‘relatedPerson’ and ‘relatedBusinessUnit’ structures contain a ‘role’ element, which can be used to convey what ‘purpose’ had previously been intended to convey.
    - ‘LoanPartyTradingInstructionOverrideStatement’
        - This statement was removed since optional settlement instruction elements now exist in all servicing and trading notifications. Related structures (i.e. ‘overrideId’) were removed.
    - ‘OutstandingContractsStatement’
        - An optional ‘DealDetails.model’ has been added to this statement, to correlate with the optional ‘dealReference’ included in the ‘facilityIdentifier’ element.

10. Changes to Models

To support uniformity and design consistency the following changes have been made to models:

- ‘DealDetails.model’
  - This model was created to reference a deal by either identifier or summary, within various statements and notifications.
  - Contains ‘dealSummary’ and ‘dealIdentifier’ elements.
- ‘LoanTradeAllocationDetails.model’
  - Revision to this model to contain the ‘LoanAllocationSummary’ element.
- ‘LoanTradingSettlementAccruals.model’
  - LoanTradingAccruingFeeAccrual
    - Changed name to LoanTradingFacilityFeeAccrual (element changed from ‘accruingFee’ to ‘facilityAccrual’)
    - Changed name of ‘accrualTypeId’ element to ‘accruingFeeTypeId’ (still complex type ‘AccrualTypeId.’
    - Changed name of ‘type’ element to ‘accruingFeeType’ (still based on the LoanTradingAccruingFeeTypeEnum).
    - Added a ‘facilityReference’ (href) into this structure.
  - ‘LoanTradingNonRecurringFee’
    - Change the name of the complex type to ‘LoanTradingNonAccruingFee’ (changed the element name from ‘nonRecurringFee’ to ‘nonAccruingFee’).
    - Moved *outside* of the model since this item did not contain any accruing fees.
    - Created and added a ‘NonAccruingFeeTypeId’ (element ‘nonAccruingFeeType’) for consistency with the ‘LoanTradingFacility FeeAccrual’ structure.
    - Change name of ‘feeType’ element to ‘nonAccruingFeeType’ and placed it within an optional choice block with ‘MiscFeeType’. That optional choice block relates to the newly created ‘NonAccruingFeeTypeId’ in a way consistent with the relationship between type and ID type within the ‘LoanTradingFacilityFeeAccrual’ structure.
  - ‘LoanTradingOutstandingAccrual’
    - Changed name of the complex type to ‘LoanTradingLoanContract Accrual’ (element name changed to ‘interest’ to ‘loanContractAccrual’).
  - ‘LoanTradingLetterOfCreditAccrual’
    - Changed element name from ‘letterOfCreditFee’ to ‘letterOfCredit Accrual’.
    - Created ‘LcFeeTypeId’ (based on ‘lcFeeTypeIdScheme’) and placed it in sequence with the new ‘LoanTradingLetterOfCreditFeeType’ (based on the ‘LoanTradingLetterOfCreditFeeTypeEnum’). The new ‘LoanTradingLetterOfCreditFeeType’ was made optional on the sequence in keeping with the design for ‘LoanTradingFacilityFee Accrual’ and ‘LoanTradingNonAccruingFee’.

- ‘PeriodWithDays.model’
    - ‘numberOfDays’ element changed from type xsd:decimal to type xsd:integer.
- ‘SimpleBuyerSeller.model’
    - Created a ‘simple’ version of ‘BuyerSeller.model’ to eliminate the account references that are not applicable to the syndicated loan market.

11. Changes to Enumerations

The following enumeration changes have been made to better support the schema:

- ‘LoanAllAssetsEnum’
    - Created this enum with a single value of ‘All.’
- ‘LoanAllTransactionsEnum’
    - Created this enum with a single value of ‘All.’

  **The above two new enumerations support the much-needed ability to explicitly indicate within the LoanPartyProfileStatement that a set of contact details and settlement instructions applies to all assets and/or all transactions.
- ‘LoanTradingLetterOfCreditFeeTypeEnum’
    - Added to support revisions to the LoanTradingSettlementAccruals.model, with values of ‘LetterOfCreditFronting,’ and ‘LetterOfCreditIssuance.’
- ‘LoanTradingNonRecurringFeeTypeEnum’
    - Changed the name of this enum to’ LoanTradingNonAccruingFeeTypeEnum’. All list items were found to be nonaccruing items, though the list was originally named “…nonRecurring…” Since ‘accruing’ and ‘nonrecurring’ are not necessarily mutually exclusive concepts, this was found to be a problematic distinction in the loan schema.
- ‘LoanTradingSettlementTaskStatusEnum’
    - Changed the name of the enum to ‘LoanTaskStatusEnum’ to make it more universally usable (as opposed to specific to the loan trading settlement scenario). Values within the enum were NOT changed.

12. Changes to Schemes

- ‘nonRecurringMiscFeeType’
    - Changed to ‘miscFeeTypeScheme’. Since this more accurately is intended to represent ‘one-off’ fee types and is a scheme, this change is appropriate to make.
- ‘personRoleScheme’
    - Added values from ‘applicablePurposeScheme’ to this scheme, along with value definitions, to support its use within the ‘communicationDetails’ element discussed above.
- ‘businessUnitRoleScheme’
    - Added values from ‘applicablePurposeScheme’ to this scheme, along with value definitions, to support its use within the ‘communicationDetails’ element discussed above.
- ‘applicableTransactionTypeScheme’

- This scheme list was updated to better reflect the structuring of substitution groups as well as business categorization of transaction types. The previous scheme list reflected the former division of servicing events based on the ambiguous collection of substitution groups that have since been revised for 5.11.

Incompatible Changes Compared to FpML 5.10 Recommendation

- Changes incompatible to FpML 5.10 are depicted above in red font.