

```
//Assume that the AbbreviationsMap is a map with the name in lowercase as key and the xml tag as value
```

```
generateXMLTag(name)
{
  if the name contains dot characters, then remove the part of the name found after the first dot
  character // this allows handling MessageDefinition name like
  CorporateActionEventProcessingStatusAdvice.002V01
  if the RepositoryConcept is a MessageDefinition, then remove the trailing version number (that is V01,
  V02,...) // as per ISO 20022 part 4
  if the AbbreviationsMap contains a key being the name in lowercase
  then
    return the value mapped by this key
  else
    camelCut the name into subparts // see description below
    for each of the subparts
      concatenate this subpart with the subsequent subparts of the name
      find the longest of those concatenations where a match is found in the AbbreviationsMap
      // for example, for the name 'ForeignExchangeTradeInstructionV01'
      // first remove the version 'V01'
      // then for the first subpart 'Foreign'
      // first check if AbbreviationsMap contains an entry for 'ForeignExchangeTradeInstruction'; it does
      not
      // then check if AbbreviationsMap contains an entry for 'ForeignExchangeTrade'; it does not
      // then check if AbbreviationsMap contains an entry for 'ForeignExchange'; it does and this maps
      to 'FX'
      if a match is found in the map // in the example we append 'FX'
      then
        append the value mapped in the map
        and continue with the rest of the name
        // in our example, for the name 'ForeignExchangeTradeInstructionV01'
        // the subpart 'ForeignExchange' was mapped to 'FX',
        // then continue with the rest of the name, that is 'Trade' and 'Instruction' (and the version
        number was removed)
        // which eventually will map respectively to 'Trad' and 'Instr'
        // And therefore, the initial name 'ForeignExchangeTradeInstructionV01' is finally mapped to
        'FXTradInstr'
      else
        report an error and stop generating the XML tag for this name
    }
}
```

```
camelCut(name)
{
  // separators are the uppercase characters, the digits, the dot, the underscore and the first character
  of the name
  // a sequence of separator characters can be a single character
  iterate over the characters of the name
  do a 'cut' if the character is the 1st or the last character of a new sequence of separator characters
  // For example
  // CSDDeposit is cut as follows [CSD, Deposit]
  // ForeignExchangeTradeInstruction is cut as follows [Foreign, Exchange, Trade, Instruction]
  // AcknowledgedMessageReference is cut as follows [Acknowledged, Message, Reference]
  // Quantity2Details is cut as follows [Quantity, 2, Details]
  // HighestPriceValue12Months is cut as follows [Highest, Price, Value, 12, Months]
}
```