OrderChangeNotif
Progress Update & Next Steps
CR 067 – Two Facets

Asynchronous Messaging Protocols

Structuring the Change
OrderChangeNotif Today

- Managing connections to 100s+ of API different endpoints is impractical & not scalable
- Dealing with API downtime and the burden of delivery (retry policies, contingency channels)

- Significant effort and responsibility for sellers to run their own API endpoints and related services / infrastructure.
Asynchronous Messaging

Delivery mechanisms considered:

- HTTP Long-Polling
- Webhooks
  - **Message Queueing**
    - STOMP
    - MQTT
    - AMQP
    - JMS
    - SQS

Persistence of one HTTP connection per client
Still requires management of API endpoints
OSI Layer 7. Text-based.
OSI Layer 4. Optimized for embedded systems.
Java-specific
Proprietary
AMQP Specs & Implementation

Two “competing” versions
• Pre 1.0 (which is de facto a standard)
• Post 1.0 since 2014 (ISO/IEC 19464)

Numerous open source implementations
• Server
  – Apache ActiveMQ (post 1.0)
  – RabbitMQ (pre 1.0)
• Client
  – Every language has an open source AMQP client implementation (post and pre 1.0)
  – Apache Qpid (post 1.0)

And also proprietary solutions & platforms…

AMQP Features
• Fully interoperable
  – Language, client, server…
• Reliable
  – Delivery / Retry-policies:
    – at-most-once
    – at-least-once
    – once-and-only-one
    – with ack…
• Security (authentication)
• Scalable
• Features: https://www.amqp.org/product/features
OCN Change Structure

- Purely technical problem area of data synchronization
- 4 x proposals considered so far - still work in progress.
- Considered OCN as a way to execute multiple sequential updates (batch) or as a single update transaction.
- Will include some additional context related to the change, e.g. REA code
- OrderHistory to be aligned with re-design of OCN
- Sticking to principles:
  - Avoid resending something the recipient already knows (RQs by reference rather than by value)
- Order versioning required
<OrderID>XB12345</OrderID>
<OrderVer>1</OrderVer>
<Element1>
  <Element1A>
    <Element1A_ID>E001</Element1A_ID>
    <Element1Aa>111</Element1Aa>
    <Element1Ab>222</Element1Ab>
  </Element1A>
</Element1>
<Element2>
  <Element2_ID>E002</Element2_ID>
  <Element2A>333</Element2A>
  <Element2A>444</Element2A>
  <Element2A>555</Element2A>
</Element2>
<Element3>
  <Element3_ID>E003</Element3_ID>
  <Element3A>
    <Element3Aa>
      <Element3Aai>666</Element3Aai>
      <Element3Aaii>777</Element3Aaii>
      <Element3Aaiv>888</Element3Aaiv>
    </Element3Aa>
  </Element3A>
  <Element3B>AAA</Element3B>
  <Element3C>BBB</Element3C>
</Element3>

<ChangeType>Update</ChangeType>
<Delete>
  <Element2>
    <Element2_ID>E002</Element2_ID>
    <Element2A>444</Element2A>
  </Element2>
</Delete>
/Add>
  <Element2>
    <Element2_ID>E002</Element2_ID>
    <Element2A>445</Element2A>
  </Element2>
</Add>

XML Segment: IDs & Values

IDs for navigation mixed in with elements/values to be updated.

IDs need to be hard-coded as “for navigation purposes only”
XML Segment + Business Context

Repeating some information from the Order to provide additional context

“Diff” required
XPaths with OCN

Separating the navigation [to where the update needs to take place] from the actual data to be updated

XPath handling already part of the same code libraries used to parse XMLs
XPaths with OCN

Separating the navigation [to where the update needs to take place] from the actual data to be updated

Could be used to update any level of the DOM
XPaths with NDC/OO

Used today in:

- All NDC RS Msgs’ Errors (since v1.1.1)
- Warnings
- Field Metadata
- Policies
- ContentSource
- Rules Metadata
- Marketing Messages
- Payload-wide / element-specific encryption (coming soon)
Thank You