Financial Services Technology Consortium Poisson paper:

Introduction:

The Financial Services Technology Consortium wishes to thank the World Wide Web consortium for allowing it to express its position in these matters, as our members who represent a significant section of members of the financial enterprise community.

In particular we wish to express our views on "the best approaches to facilitate the processing of business transactions and interactions with systems that pre-date the Web, and address the need to interconnect intranet and/or extranet servicesusing Web technologies."

Our position:

Great progress has been made since the last workshop on web services use to industry. In particular the ability to do interoperable computing using SOAP, WS-Addressing, WS-Security etc, These are moving forward at a good pace, so long as one speaks about the HTTP, and HTTPS protocols.

- When one thinks about using other Internet protocols such as FTP, SMTP, etc virtually no progress has been made. The financial services industry, which has a deep legacy of dealing with batch driven computing in order to do many of its back office functions as well as real time computing in order to do many of its trading, and intraday functions needs to insure that an architecture that supports these batch processes, and real time processes exists on the internet, and within the messaging infrastructure of the internet.
- Retail Banks, insurance companies, investment banks, and the businesses that service the financial markets such as financial news business entities must often deliver massive amounts of information to partners, and that data must be actionable in near real time by the consumers of the information embedded within it. Thus we often need bridging technologies to bind a batch process to a real time process.
- We work in a highly distributed environment where it is often impossible to work in a point to point manner. Rather we work within a distributed architecture in which messages often pass through many intermediaries (proxies, both active and passive) that serve vital functional and non-functional roles within the processing that must take place in order for the content of a message to be successfully processed.
- The messages we send more often then not do change the state of the systems that pass them and are thus transactional in nature. Thus the architecture we seek from our web service architecture must be capable of reliably changing state in a predictable manner when passing through complex architectural paths.
- We must often broadcast information that changes the state of many systems, and must do so within precise time frames. These broadcasts must often be reliable. The process utilized is that Market data is, often, broadcast from the supplier (UDP) and then retransmitted to internal systems via TCP for better reliability.
- We use web service architecture
  - To get to market faster with business innovation

    - To encapsulate legacy systems so as to be able to methodically replace them.
    - To allow for one service to be used for many purposes
    - To bridge between heterogeneous language and platform environments.
    - To bridge one enterprise to another.
- We wish for the work that takes place standardizing the protocols that enable this to move more rapidly.
- Many of us are forced to use proprietary solutions when we need to pass reliable messages in a manner that is more then trivial.
- Virtually all of us use proprietary solutions when we wish to broadcast market data related to prices.
- Management of our systems using internet architecture that is standards based is virtually impossible at this point. Though work is taking place to standardize this, the work is taking place slowly and this slowness is inhibiting our ability to embrace a standards based architecture.
- We want to build reference architecture within our sector, and yet we do not see a suitable, actionable reference architecture to use as a platform to do this work as it pertains to our industry.
- The space of choosing a standard, as one to embrace is confusing. The landscape sometimes seems to change without solid engineering rhyme or reason, rather it seems to change in order to meet the commercial interests of particular sets of vendors.
    - We site the management protocols and the reliability protocols as examples of this.
- We need to bridge wire level protocols such as HTTP and FTP to API level protocols such as JMS, and M Queue

Recommendations

1. FSTC members have a strong desire to see specific specifications, like WS-Transactions and WS-ReliabileMessaging, get approved and into interoperability evaluation & testing. The Financial services industry needs standards that support long-lived transactions and a transport infrastructure that supports guaranteed message delivery.
2. Establish bindings for FTP
3. Leverage the features of message oriented (MOM) and RPC-based middleware that almost all or our legacy systems are built on.
4. Establish a reference architecture that accounts for:
    - Requirements
    - Delegations of authority to components that interface in accordance with standardized wire, and in memory, models that are unambiguous
    - Interfaces both at a wire level and an in memory API level where appropriate. Bindings to specific languages might be out of scope for this, but a clear model for

passing XML documents and delegating authority to subordinate tasks must be in scope.

4. Establish policy languages for various non functional and functional capabilities such as 'quality of service,' audit, & routing
5. Establish behavior for service intermediaries with the same degree of excellence as did HTTP 1.1.
6. Insure backward compatibility to the currently, generally implemented web service standards.
7. Insure compatibility between 'REST' services that are built in a more adhoc nature then those that use the WS* stack.
8. There is a need for an independent third party to provide interoperability certifications. Today it is a vendor self-assessment. There needs to be a set standard interoperability test suites developed. If there is no appetite for this at this forum, then the financial services industry should develop and start using it's own. We should insist on achieving the widest range of interoperability possible while always trying to prevent vendor "lock-in" caused by feature rich IDEs and development tools that can generate code that leads to interoperability problems.