SCA Event Processing and Pub/Sub :

An Implementation Perspective

History

- Implementation work started: mid-2006
- Event proposal presented to OSOA: early-2007
- OSOA debated 5 different proposals & produced a consolidated spec: April 2009
- OASIS Assembly TC Submission: May 2009
- First event-focused f2f: November 2009

Motivation

- Application Integration
- Integration of SCA components with existing infrastructure and interaction models
- Leverage well-accepted pub/sub style of organizing components
- Decoupled producers/consumers
- Components interacting via events

Motivation(2)

- SCA components consume events by invoking SCA services
- SCA services process requests by raising events

Scenarios

- Based on current implementation experience and user feedback
- Scenario 1: Overdrawn bank account
  - Account overdrawn event
  - Component A applies an interest rate
  - Component B sends an alert (email/SMS)
  - Component C freezes the account
  - In the future component D may be added that detects fraud
  - Consumers outside of an SCA domain may be interested in the event
  - Producers outside of the an SCA domain may be raising the event

Scenarios (2)

- Scenario 2: Sales event processing
  - Orders come in over the web and this results in an event
  - Orders triggers processing at various consumers including order processing components: shipping, credit card verification, inventory check etc
  - Results in additional events being raised (inventory update, shipping notice etc)
  - Additional consumers may be added in the future: marketing, JIT inventory management
  - Additional producers may be added in the future

Scenarios (3)

- Scenario 3: New employee provisioning
  - New employee gets added or removed
  - Triggers processing at various consumers: accounts provisioning, payroll, facilities, directory update etc
  - Consumers outside of the SCA domain are interested
  - Producers outside of the SCA domain produce the events

Implementation (Non-)Requirements

- Requirements gathered from about four years of implementation experience and user feedback
- JMS is the implementation technology
- Channels not scoped to be SCA-only
  - External producers can raise events on an SCA channel
  - External consumers can consume events from the SCA channel
- Web services:
  - SOAP packaging not used for event messages, so WS-* specs have no use
  - Outside the SCA boundaries WS-* spec such as WS-Eventing used for subscription management
- Filtering at channels not implemented
  - Channel does not reject messages
  - Trees falling down and no one around to hear it, is allowed

Implementation (Non-)Requirements (2)

- Partitioning mechanism (aka channels) not found to be useful
  - Event types and other filters are sufficient to create virtual partitions
  - On deployment, knowledge of all the participants allows for optimization, load-balancing etc
- JMS administration is painful/confusing for users
  - SCA composites should stay away from similar configuration info unless absolutely necessary
  - SCA runtimes should be allowed to provide the necessary “magic” for creating appropriate JMS destinations without user input
  - JMS topic and SCA channel mapping should be decoupled
  - SCA runtime should be allowed to change the underlying JMS configuration and/or mapping dynamically

Implementation (Non-)Requirements (3)

- Event source/destinations used as global resources
  - Channels viewed as global rather than local
  - Access control is provided by deploying appropriate security apparatus rather than information hiding or using scopes
  - Consumers/Producers can connect to the same set of channels regardless of their position in the composition hierarchy
  - No local SCA channels
  - Promotion of channels adds to the administration pain-points and confuses users
  - Event visibility controlled by topic/event type or filters/security/connectivity not by composition scope
  - Recursive composition used for overriding config not visibility of events