OASIS Business Transaction Protocol:

Multi-party Coordination for Commercial Collaborations

alastair.green@choreology.com
peter.furniss@choreology.com
tony.fletcher@choreology.com

February 2002
OASIS Business Transactions TC

Initiated mid-January 2001

- BEA, Bowstreet, Sun, Interwoven

Formed mid-March

- Three initial submissions: BEA, HP, Choreology
- Six face-to-face meetings
- HP, Oracle, Sybase, Entrust, IONA, Choreology, Talking Blocks, SeeBeyond, Systinet
- www.oasis-open.org/committees/business-transactions

Targetting OASIS Committee Specification
Goals

Inter-organizational multi-party coordination

Targetting Web Services, but not only WS

Accommodate Long-running Transactions
Requirements

Interoperation
- Using XML, over multiple communications protocols

Coordination of autonomous parties
- Relationships are governed by contracts, rather than the dictates of a central design authority

Drop less ACID
- Multiple possible successful outcomes to a transaction
- Relaxed isolation, volatile results

Discontinuous service
- Work unit lifespans exceed sub-system MTBFs
**Business Transactions and Business Process**

BTP complements BP/Collaboration frameworks

- A coordinated, mutually understood outcome requires
  - Special messages and acknowledgements
  - Consistent, durable record of decisions
  - Asynchronous failure recovery operations
- These features are tricky, error-prone and intrusive

“Build or buy”

- BTP lets business people concentrate on business process
- Puts housekeeping work in the background
- Minimizes application exchanges
  - Reduces complexity of collaborative process schemas or scripts
  - Reduces conformance testing
- Deploy new trading protocols or conventions more quickly
BTP Feature Stack

- Trading Community Extensions
- Cohesion Composition
- Atom Coordination
- Service-defined Operation Groups
- Participants
Atoms and Cohesions

BTP uses a two-phase outcome coordination protocol to create atomic effects (results of computations).

BTP permits the composition of atomic units of work into cohesive business transactions (cohesions) which allow application selection of which work units will be confirmed (or cancelled)

Atoms are cohesions where the underlying work units are either all confirmed, or are all cancelled
Atom Coordination: The key BTP roles

Client side

Application

Initiator/Terminator

Factory

Transaction Decider (Composer or Coordinator)

Service side

Service

Enroller

Participant
Ask Factory to Create top coordinator

Application

Client side

Service side

Protocol

Client

Initiator/Terminator

BEGIN / BEGUN\&CONTEXT

Factory

creates

Coordinator

Service

Enroller

Participant
Send Application Message with CONTEXT

Client side

Application

Protocol

Factory

Initiator/Terminator

Coordinator

Service side

CONTEXT

Service

Enroller

Participant
Service creates and causes ENROL of Participant
Service-defined Operation Groups

Services provide forward operations

- Application computations
- Must log information needed for confirm/cancel

Services use Participants to supervise outcome

- Result of group of forward operations is either confirmed …
- Or counter-effected by cancellation

Cancellation behaviour is service-defined
Send Application Reply with CONTEXT_REPLY
Two-phase Confirmation: PREPARE phase
Two-phase Confirmation: Outcome phase
REQUEST_CONFIRM Reply

Client side
- Client
- Initiator/Terminator
- Factory
- Coordinator

Service side
- Service
- Enroller
- Participant

Protocol
- TRANSACTION_CONFIRMED
Cancellation

Example showed CONFIRM/CONFIRMED

Terminating application can also issue
CANCEL_TRANSACTION

- Or the CONFIRM_TRANSACTION can fail, leading to TRANSACTION_CANCELLLED
- CANCEL/CANCELLED exchange with Participant

Operation Group = Forward Ops + Participant

- Participant responsible for cancel or confirm
- Cancel can be compensation of committed DB transaction
- Or rollback of uncommitted DB updates
Operation Groups: Provisional effect and Counter-effect

- Forward Operations create a provisional effect …
- … and durably record information needed by Participant
- Participant uses log to perform final effect or to counter-effect
The Counter-effect Contract

Provisional effect and Counter-effect

- Final effect means “complete effect, and throw away log”
  - Final effect is action on CONFIRM
- Counter-effect means “reverse effect, and throw away log”
  - Counter-effect is action on CANCEL

Counter-effect Contract

- Each Operation Group can define counter-effect differently
- Anything from pure inversion to “we’ll take 75% cancellation fee”
- Default “counter-effect contract” in specification
  - As close to inverse operation as possible
  - Expected to be overridden in many cases
2PC ≠ ACID

Example #1: Compensation strategy

- Provisional Effect includes committed database updates or message enqueues
  - E.g. debit credit card account
- Final effect is no-op
- Counter-effect involves compensatory action
  - E.g. contra-credit credit card
  - In whole or in part depending on business contract

Example #2: XA RM

- Provisional Effect posits but does not commit database updates or MQPUTs
- Final effect ⇒ invoke xa_commit
  - Throw away RM undo logs
- Counter-effect is to invoke xa_rollback
  - Process RM undo logs
Coordination Hub: An alternate topology

Client

Application

Protocol

Coordination Hub

Initiator/Terminator

Transaction Decider (Composer or Coordinator)

Factory

Service

Enroller

Participant
Superior-Inferior Relationship - outcome

Superior
- PREPARE
- CONFIRM
- CANCEL

Composer
Coordinator
Sub-composer
Sub-coordinator

Inferior
- PREPARED
- CONFIRMED
- CANCELLED

Sub-composer
Sub-coordinator
Participant
Superiors and Inferiors

Superiors transmit outcomes

- Composer of a Cohesion (spans multiple Atoms)
  - Can send CONFIRM to some Atoms, and CANCEL to others
- Coordinator of an Atom
  - Sends same outcome to all of its Inferiors (Sub-coordinators, Participants)
- Sub-composer and Sub-coordinator
  - Act as Inferior to parent node in transaction tree
  - Act as Superior to children

Inferiors “vote” on the outcome

- Sub-coordinators and Sub-composers
  - Act as intermediaries connecting decision maker to participants
- Participants
  - Leaves of the tree: cancel or confirm application (forward) operations
Transaction Tree

Superior
[e.g. Coordinator]

Inferior/Superior
[e.g. Sub-coordinator]

Inferior
[e.g. Participant]
Terminators and Deciders

**Terminators - volatile**

- Application function - requests top Superior to seek to confirm its Inferiors

**Superiors - persistent**

- Log destinations of CONFIRM messages
- Can be contacted by Inferiors after a crash
- Able to replay the decision (resend the CONFIRM s)

**Decider - top superior**

- If top superior is asked to confirm but cannot log confirm decision it must cancel
  - Top superior can contradict Terminator
  - Ultimate decision maker holds outcome decision durably
Control and outcome relationships

- Client
- Service
- Protocol
- Application
- Coordination Hub
- Initiator
- Terminator
- Factory
- Decider
- Superior
- Enroller
- Inferior
- Participant
Failure Recovery

Protocol incorporates recovery after failure
- Superior system, Inferior system or network may fail
- Must try to re-establish Superior-Inferior relationship
- Allows outcomes to be replayed

Standard “presumed abort” protocol
- No durable record (log) equals absence of decision
  - Default decision (in absence of evidence to contrary): CANCEL

Bi-directional recovery initiation
- Superior can attempt to contact logged In inferiors
- Inferior can attempt to contact logged Superiors
Cohesions: The Concept

- **Cohesion Terminator**
- **Superior Composer**
- **Inferior Coordinators Superior**
- **Inferiors Participants**
- **Goods Atom**
- **Shipping Atom**

- **Buyer**
- **Services**

#1 and #2 are connected to the other components.
Cohesions: PREPARE both Atoms

Superior Composer

Inferior Coordinators Superior

PREPARE_INFERIORS #1, #2

#1

#2

PREPARE

PREPARE

PREPARE

PREPARE

Buyer

Services

Inferiors Participants

Goods Atom

Shipping Atom
Cohesions: both are PREPARED
Cohesions: CONFIRM #1 (⇒ CANCEL #2)

CON Firm_TRANSACTION #1

 CONFIRM

#1

CANCEL

#2

CANCEL

CANCEL

CANCEL
Cohesions: #1 CONFIRMED, #2 CANCELLED

**Superior**

**Inferiors**

- **INFERIOR STATUSES**
  - #1 CO'd
  - #2 CA'd

- **Coordinators**

- **Superior Terminators**

**Services**

**Buyer**

**Outline**

- **CONTRACTS**
  - CONFIRMED
  - CANCELLED

**Shipping**

**Goods Atom**

**Inferiors Participants**

---

Choreology® The Interplays of Commerce
Cohesions rely on “Open-top” Coordinator

CONFIRM / CONFIRMED

PREPARE / PREPARED

CANCEL / CANCELLED

CONFIRM_TRANSACTION

INFERIOR_STATUSES

#1 CO’d

#2 CA’d

#1

#2
Long-running features

Superior and Inferior negotiate time band

- Inferior threatens to auto-cancel or confirm after (at least) \( n \) seconds
  - Superior qualifies the PREPARE message
  - Inferior qualifies the PREPARED message
- Prevents coordinator hogging service provider’s resources
- Prevents service wasting coordinator’s time
- Independent organization likely to demand this power

Active and prepare phase recovery

- Allows Superior/Inferior re-synch after failures
  - Standard 2PC protocols allow recovery only after prepared
- BTP treats failure as a potential interruption
  - Standard 2PC protocols treat any failure as cause to cancel
All BTP messages are XML documents

- Can be compounded for optimization
- "One-shot requests": only 2 WAN messages instead of 6
  - Application response + ENROL/PREPARE
- "One wire" application topologies
  - All traffic between two business entities over a single, authenticated link

Binding of abstract set to SOAP

- Defined in the specification

Other bindings are possible

- To any underlying communications protocol stack
“Trading Community” Extensions

QUALIFIERs can be embedded in messages
- Some are defined within BTP specification
- Implementers/applications can define their own

Allows “trading communities” to define extensions to protocol messages
- E.g. Could be used by trading parties to add security data
- E.g. Could be used by implementer to add full nested transactions

Allows application data to travel with protocol
- Example: confirm a two-way quote
- Must include “buy” or “sell” in CONFIRM
Do Business Together with Business Transactions

Ensuring Collaboration for XML Services