OASIS Business Transactions Technical Committee
Submission to the Inaugural Meeting on 13 March 2001, at San Jose, CA

Scope of a Cohesion Protocol specification

Summary. *Long-running "business transactions" which may be processed by discrete organizations across the public internet differ from classical atomic transactions in requiring increased protocol security and interoperability, and relaxable atomicity, isolation and durability properties. A protocol is required which is independent of communications mechanism, is capable of supporting fully ACID transaction processing, yet is also capable of supporting different AID qualities of service. Such a protocol would provide "appropriate transactionality" to applications. "Cohesive" actions (cohesions) could be processed as a superset of atomic actions, thus enabling a clean integration of legacy transactional resources and services, when appropriate.*

Alastair Green, Peter Furniss, 12 March 2001
Choreology Ltd[1].13 Austin Friars, London EC2N 2JX. Tel +44 20 7670 1679

`<first>.<last>@choreology.com`

[1] Choreology Ltd is a recently-formed company, founded by a small group of architects, managers and developers who formerly worked for Bluestone/HP Arjuna Lab (the home of the first commercial pure Java OTS implementation). Our Technical Director, Peter Furniss, is Maintenance Rapporteur for OSI/TP.

Introduction

This paper was conceived prior to the distribution of other submissions to the OASIS Business Transactions Technical Committee.

Scope

It is widely perceived that inter-organizational long-duration business transactions require a new protocol or protocols to assist applications in providing reliable service and consistent results. However, the complex logic of extended machine-to-machine conversations will necessarily be an application concern, often assisted by a business process manager.

This paper is intended to help in drawing a reasonable boundary between protocol (the concern of the BT TC) and process (the concern of other standardization efforts or of the application). Our principal guide is that *a BT protocol should not be aware of application control flow, application message content, or operation algorithms and effect.* This criterion excludes business process definition and trading protocols (such as ebXML, in that role).

We believe that this leaves three distinct problems which are the potential concern of the Technical Committee: communications, interoperability/security, and ACIDity. In this initial submission we restrict our comments largely to the latter two problems.

Requirements

We have listed below several potential requirements for the protocol specification to address. We have not tried to rank these requirements.

Fundamental requirements

> *Prefatory note:* Applications are conventionally perceived as systems owned by one organization. In the following discussion we recognize that an application may be a cooperative effort conducted by sub-systems with different organizational ownership directed towards a common business outcome. An example would be a long-running interplay between two parties to establish contact, negotiate trading terms and capabilities, negotiate contract terms, order, invoice and settle.

At the most fundamental level, applications carry out some set of operations (computations). In a web services environment, where the client and the operation are potentially in different organizations, it is useful to define operation groups, where all forward operations in the group can be can be backed out by a single reverse operation. To be precise, *the reverse operation will attempt to make good the effect of unwanted forward operations*, where "make good" has meaning for the application: it may not be possible to simply process an inverse effect:

Requirement #1: **Operation groups with reverse operations**

In addition, each operation group must have a mechanism for ensuring that the group's completed operations constitute an atom (are all processed to completion):

Requirement #2: **Operation group atomicity**

It is worth noting that operation groups may only contain one operation (which is likely in a pure web service, and unlikely in a conventional database application).

In practice applications compose operation groups into "business transactions": sets of operations that once completed will effect a useful piece of work from a business standpoint. The term "business transactions" is heavily overloaded: we will use the term *action* to describe a "set of intended operation groups". The membership of the set must be decided by the application:

Requirement #3: **Action demarcation (addition and removal of operation groups)**

It is frequently the case that an action may be cancelled. Another way of stating this is that the effect of an action can be reversed, irrespective of how many of its operations have actually completed.

If an application has to satisfy the reversal requirement without protocol assistance then it is forced to take on the responsibility for tracking all related, reversible operations, and ensuring that they get reversed, including in the event of process and communications failures. This want is difficult to satisfy, justifying the existence of a transactional protocol.

Requirement #4: **Action reversal**

An application should further be able to control the subsets of operation groups within an action that are valid outcomes of the action. In a classic atomic action (e.g. an X/Open DTP or OTS transaction) there can only be one valid outcome. In a long-running transaction, where resources are often being assembled in stages of a business process, it is necessary to treat partial success as a useful, valid outcome. This implies

Requirement #5: **Multiple valid action outcomes**

The ability to tolerate partial failure implies that the application must be allowed to remove or "knock out" operation groups that become useless or ineffectual from an application standpoint. This facility allows, for example, reservations to be cancelled, and replaced by alternatives. Such a removal, or substitution, only enhances the validity of the action. This requirement has already been stated as part of Requirement #2: **removal of**

operation groups. It should be noted that any part of the application may want to remove an operation group.

In a loosely-coupled long-duration action, time plays a key element in deciding which operations participate, and how viable the action as a whole is. Timeouts can be negative or positive in their effect (rollback or keep, commit or forget). From this flows

> *Requirement #6:* **Positive/negative action timeout**
>
> *Requirement #7*: **Positive/negative operation timeout**

Serializability and persistent effect (isolation and durability) requirements

Long-running business transactions may cause an unacceptable level of serialization if they cause other actions to be blocked. On the other hand they may require that serializability is imposed at particular points. A scheme is needed whereby operations can use whatever cooperative blocking mechanisms make sense from an application standpoint. As operations are written separately from the actions that compose them, and their very existence may not be known in advance to an application, the isolation characteristics of operations may have be discoverable by applications.

> *Requirement #8:* **Operations can use (and can advertise) differing isolation levels (degrees of blocking)**

The effect of operations may need to be persistent (survive over process failure), or it may not. Some actions coordinate a related set of operations on volatile storage (e.g. provisioning of network elements). The ability to reverse such an action is very valuable, but there is no need to associate persistent effect with its operations. Any situation where the volatile state can be reconstructed after failure, or is of evanescent value, generates the need for

> *Requirement #9:* **Operations can use (and can advertise) differing durability levels (degrees of persistence)**

Valid actions may require consistent or interoperable levels of isolation and durability across all operations. This implies that discovery may need to be followed by negotiation, and that applications may be able to select levels from sets of levels offered by operations:

> *Requirement #10:* **Application/operation negotiation over isolation and persistence levels.**

Interoperability and portability requirements

Inter-organizational business interchanges of necessity imply distribution of processing. Such interchanges are likely to occur across a network of processing nodes where the arcs that join the nodes are created using multiple communications protocols (SOAP, IIOP, HTTP, SMTP etc.). Any useful protocol must therefore support

Requirement #11: **Communications-protocol independent distribution**

Which must occur across organizations, where operation implementations are opaque. From this follows mandatorily:

Requirement #12: **Interoperability**

An interoperable protocol will be based on agreed network message formats for distributed processing and clear contractual commitments on the parties involved in requesting or receiving the invocation of a computation.

It is assumed as an overriding requirement that message formats will be described in terms of XML schemas, capable of validation on despatch and receipt:

Requirement #13: **XML schema-based protocol message formats**

Note: At this point we are not concerned with the issue of transaction domain interoperability, i.e. the problem of how to talk to entities using other transactional protocols. We address this problem under the heading "Legacy Integration Requirements" below.

Security requirements

The inter-organizational provision also raises the need for authentication. We assume that application entity authentication and consequent authorization (permissioning) is the province of the application. Here we are concerned with the fact that the protocol itself will pass messages across trust boundaries. This is an issue which prior transactional protocols have often not addressed. At the same time, security should be appropriate to the task at hand. This implies

Requirement #14: **Configurable protocol authentication**

In addition, in a conversational mode it is possible that there will be no clear owner of the transaction in the sense of a single application element solely permitted to terminate the transaction. Application elements may or not be permitted to unilaterally join transactions.

Requirement #15: **Access control for joining and terminating transactions**

Legacy integration requirements

There are large numbers of systems which use conventional transactional protocols. The most prevalent are X/Open DTP (Tuxedo/Encina-style TPMs in conjunction with support for XA by major database and messaging vendors); JTA/JDBC 2.0 Standard Extensions, which allow XA resources to be utilized from Java applications, and OTS/JTS, the CORBA-based transaction service.All of these approaches integrate XA-compliant resource managers as participants in distributed transactions. A new extended transaction protocol should follow this pattern, thereby enabling very significant application integration:

Requirement #16: **XA integration**

Integration with other transaction services is, in our opinion, best addressed by protocol converters or bridges. The specification of a standard resource registration message set or interface will assist this greatly:

Requirement #17: **"Resource" registration scheme**

Such a scheme will coincidentally assist in the creation of bespoke and novel participants (including all "one-shot" web service operations), which will, we hope, increase the acceptability of the standard for application designers.

Note on Terminology

The terms "transaction" and "business transaction" are both used in numerous (and often overlapping or contradictory) senses. To avoid the connotations and imprecisions that this can cause, we use the term *cohesion* or *cohesive action* to describe an action (as previously defined: a set of related intended operation groups) which is capable of supporting varying degrees of atomicity, isolation and durability. A fully ACIDic atomic action in this scheme is a special case of a cohesive action. [We have also toyed with the terms "molecular action" or "molecule" in place of cohesive action and cohesion.]

Cohesive actions are trees which grow by operation group insertion and shrink by operation group removal; the tree forming the identity of the cohesion.

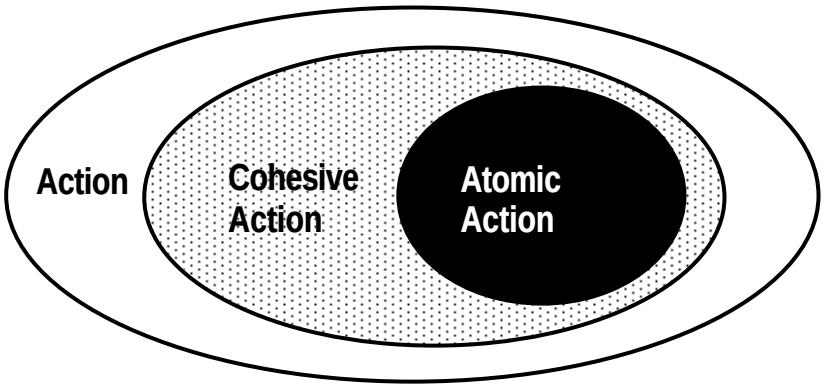
It would be good to find a less loaded term for "operation groups" than "participant" or "resource". Participant is sometimes used in contradistinction to sub-coordinator, which creates an artificial division (sub-coordinators play a dual role, that of participant and that of coordinator). The term "resource" frequently connotes "XA-compliant database", or more generally, a "resource manager" outside the application.

Instead, we wish to emphasize that operations upon participants are opaque computations (from the standpoint of a transactional protocol).

Finally, we assume that a *cohesion service* will be available to support protocol processing in any application element that uses the protocol. This service will ensure that cohesion completion is correctly coordinated, i.e. it will play the role of *coordinator* on behalf of each cohesion.

The Venn diagram below shows that an action may be a cohesive action (cohesion), and that a cohesive action may be an atomic action.

Set relationship of actions, cohesive actions and atomic actions



These terms can be equated to several approximate conventional usages, as shown in the following table:

Term	*Conventional or loose equivalents*
Action	Business process Workflow Choreography Business transaction Interchange, interplay Transaction (legal, economic) Long-running transaction (à la Biztalk) Transaction (EDI)
Cohesive Action, Cohesion	Long-running transaction Extended transaction Advanced model transaction Business transaction B2B transaction
Atomic Action	Transaction System transaction ACID transaction

Some features of a Cohesion Protocol

In this final section we make some comments on salient conclusions we have reached in our work on this area to date.

Two-phase commit does not imply strict two-phase locking

There is a widespread tendency to identify the two-phase commit protocol (prepare, commit, rollback) with serializable concurrency control in resources used by a transaction, frequently implemented by strict two-phase locking, where locks are acquired as the transaction progresses, held for its duration, and only dropped on its completion.

This assumption tends to obscure the underlying purpose and justification of the two-phase commit protocol itself. The 2PC protocol on its own cannot (and is not intended to) guarantee ACID properties. It is concerned with ensuring the readiness to affirm or cancel of all participants, in a multi-participant action. This contributes to the ability to guarantee ACIDity. This protocol can also be used to coordinate actions with less stringent isolation and durability properties.

Combining cohesions and atomic transactions

We believe that there is no inherent reason why cohesions (sub-ACIDic actions) and ACIDic actions cannot be supported by the same coordination protocol. We also believe (subject to certain application disciplines) that atomic actions can be subordinated to cohesive actions. This feature would greatly enhance the value of a new standard protocol, because of the ability to integrate existing transaction systems (domains, resources) into the newer world.

Qualities of Service

It may be useful to view atomicity, isolation, durability and security as configurable Qualities of Service; and to construct a framework which permits automatic negotiation and mutual adjustment, and which allows application elements to determine whether the reliability/integrity guarantees that other parties (trading partners) offer are sufficient for business to be done.

Peer-to-peer

Some recent transactional protocols (e.g. OTS/JTS) assume a request/response model. This model is frequently used, and simplifies some aspects of the transactional protocol. The ability to use such a model should be combined with the ability to select a freer, peer-to-peer based approach, particularly given the discrete organizational units that are likely to be involved in e-commerce choreographies.