1 Organization for the Advancement of Structured Information Systems

2 # Business Transaction Protocol

3

4

5 ## An OASIS Committee Specification

6

7 Version 1.0 *[0.9]*

8 DD Mmm 2001 *[24 October 2001]*

9

10

Working draft 0.1 (pre-London)	14 June 2001
Working draft 0.2 (London)	18 June 2001
Working draft 0.3a (circulated)	12 July 2001
Working draft 0.3b (not circulated)	17 July 2001
Working draft 0.3c (circulated)	20 July 2001
Working draft 0.4 (circulated; incorporates PRF material)	25 July 2001
Working draft 0.5 (uncirculated)	8 August 2001
Working draft 0.6 (State tables)	31 August 2001
Working draft 0.7 (revised abs msgs) – (not circulated)	28 September 2001
Working draft 0.72 (completed abs msg revsn 0.7)	3 October 2001
Working draft 0.80 – full scope, PRF handback	18 October 2001

11

Pre-final Draft 0.9	**24 October 2001**

12

13

Copyright and related notices

Acknowledgements

Employees of the following companies participated in the finalization of this specification as members of the OASIS Business Transactions Technical Committee:

> BEA Systems, Inc.
> Bowstreet, Inc.
> Choreology Ltd.
> Entrust, Inc.
> Hewlett-Packard Co.
> Interwoven Inc.
> IONA Technologies PLC
> SeeBeyond Inc.
> Sun Microsystems Computer Corp.
> Talking Blocks Inc.

The primary authors and editors of the main body of the specification were:

> Alex Ceponkus (alex@ceponkus.org)
> Peter Furniss (peter.furniss@choreology.com)
> Alastair Green (alastair.green@choreology.com)

Additional contributions to its writing were made by

> Sanjay Dalal (sanjay.dalal@bea.com)
> Mark Little (mark_little@hp.com)

We thank Pal Takacsi-Nagy of BEA Systems Inc for his efforts in chairing the Technical Committee, and Karl Best of OASIS for his guidance on the organization of the Committee's work.

In memory of Ed Felt

Ed Felt of BEA Systems Inc. was an active and highly valued contributor to the work of the OASIS Business Transactions Technical Committee.

His many years of design and implementation experience with the Tuxedo system, Weblogic's Java transactions, and Weblogic Integration's Conversation Management Protocol were brought to bear in his comments on and proposals for this specification.

He was killed in the crash of the hijacked United Airlines flight 93 near to Pittsburgh, on 11 September 2001.

Typographical and Linguistic Conventions and Style

The initial letters of words in terms which are defined (at least in their substantive or infinitive form) in the Glossary are capitalized whenever the term used with that exact meaning, thus:

> Cancel
> Participant
> Application Message

The first occurrence of a word defined in the Glossary is given in bold, thus:

> **Coordinator**

Such words may be given in bold in other contexts (for example, in section headings or captions) to emphasize their status as formally defined terms.

The names of abstract BTP protocol messages are given in upper-case throughout:

> BEGIN
> CONTEXT
> RESIGN

The values of elements within a BTP protocol message are indicated thus:

> BEGIN/atom

BTP protocol messages that are related semantically are joined by an ampersand:

> BEGIN/atom & CONTEXT

BTP protocol messages that are transmitted together in a compound are joined by a + sign:

> ENROL + VOTE

XML schemata and instances are given in Courier:

```
<btp:begin> ... </btp:begin>
```

Illustrative fragments of code in other languages, such as Java, are given in Lucida Console:

```
int main (String[] args)
{
}
```

Terms such as MUST, MAY and so on, which are defined in RFC [TBD number], "[TBD title]" are used with the meanings given in that document but are given in lowercase bold, rather than in upper-case:

147   An Inferior **must** send one of RESIGN, PREPARED or CANCELLED to its
148   Superior.
149
150

Contents

Part 1. Purpose and Features of BTP

Introduction

This document, which describes and defines the Business Transaction Protocol (BTP), is a Committee Specification of the Organization for the Advancement of Structured Information Standards (OASIS). The standard has been authored by the collective work of representatives of ten software product companies (listed on page 3), grouped in the Business Transactions Technical Committee (BT TC) of OASIS.

The OASIS BTP Technical Committee began its work at an inaugural meeting in San Jose, Calif. on 13 March 2001, and this specification was endorsed as a Committee Specification by a [*** unanimous] vote on [*** date].

BTP uses a two-phase outcome coordination protocol to create atomic effects (results of computations). BTP also permits the composition of such atomic units of work (atoms) into cohesive business transactions (cohesions) which allow application intervention into the selection of the atoms which will be confirmed, and of those which will be cancelled.

BTP is designed to allow transactional coordination of participants which are part of services offered by multiple autonomous organizations (as well as within a single organization). It is therefore ideally suited for use in a Web Services environment. For this reason this specification defines communications protocol bindings which target the emerging Web Services arena, while preserving the capacity to carry BTP messages over other communication protocols. Protocol message structure and content constraints are schematized in XML, and message content is encoded in XML instances.

The BTP allows great flexibility in the implementation of business transaction participants. Such participants enable the consistent reversal of the effects of atoms. BTP participants may use recorded before- or after-images, or compensation operations to provide the "roll-forward, roll-back" capacity which enables their subordination to the overall outcome of an atomic business transaction.

The BTP is an interoperation protocol which defines the roles which software agents (actors) may occupy, the messages that pass between such actors, and the obligations upon and commitments made by actors-in-roles. It does not define the programming interfaces to be used by application programmers to stimulate message flow or associated state changes.

The BTP is based on a permissive and minimal approach, where constraints on implementation choices are avoided. The protocol also tries to avoid unnecessary dependencies on other standards, with the aim of lowering the hurdle to implementation.

Development and Maintenance of the Specification

For more information on the genesis and development of BTP, please consult the OASIS BT Technical Committee's website, at

As of the date of adoption of this specification the OASIS BT Technical Committee is still in existence, with the charter of

- ❑ maintaining the specification in the light of implementation experiences
- ❑ coordinating publicity for BTP
- ❑ liaising with other standards bodies whose work affects or may be affected by BTP
- ❑ reviewing the appropriate time, in the light of implementation experience and user support, to put BTP forward for adoption as a full OASIS standard

If you have a question about the functionality of BTP, or wish to report an error or to suggest a modification to the specification, please subscribe to:

bt-spec@lists.oasis-open.org

Any employee of a corporate member of OASIS, or any individual member of OASIS, may subscribe to OASIS mail lists, and is also entitled to apply to join the Technical Committee.

The main list of the committee is:

business-transaction@lists.oasis-open.org

Overview of the Business Transaction Protocol

A Business Transaction is a consistent change in the state of a business relationship between two or more parties. BTP provides means to allow the consistent and coordinated changes in the relationship as viewed from each party.

BTP assumes that for a given business transaction state changes occur, or are desired, in some set of parties, and that these changes are related in some business-defined manner.

Typically business-defined messages ("application messages") are exchanged between the parties to the transaction, which result in the performance of some set of operations. These operations create provisional or tentative state changes (the transaction's effect). The provisional changes of each party must either be confirmed (given final effect), or must be cancelled (counter-effected). Those parties which are confirmed create an atomic unit, within which the business transaction has a consistent final effect.

The meaning of "effect", "final effect" and "counter-effect" is specific to each business transaction and to each party's role within it. A party may log intended changes (as its effect) and only process them as visible state changes on confirmation (its final effect). Or it may make visible state changes and store the information needed to cancel (its effect), and then simply delete the information needed for cancellation (its final effect). A counter-effect may be a precise inversion or removal of provisional changes, or it may be the processing of operations that in some way compensate for, make good, alleviate or supplement their effect.

To ensure that confirmation or cancellation of the provisional effect within different parties can be consistently performed, it is necessary that each party should

> ❏ determine whether it is able both to cancel (counter-effect) and to confirm (give final effect to) its effect

> ❏ report its ability or inability to cancel-or-confirm (its preparedness) to a central coordinating entity

After receiving these reports, the coordinating entity is responsible for determining which of the parties should be instructed to confirm and which should be instructed to cancel.

Such a two-phase exchange (ask, instruct) mediated by a central coordinator is required to achieve a consistent outcome for a set of operations. BTP defines the means for software agents executing on network nodes to interoperate using a two-phase coordination protocol, leading either to the abandonment of the entire attempted transaction, or to the selection of an internally consistent set of confirmed operations.

BTP centres on the bilateral relationship between the computer systems of the coordinating entity and those of one of the parties in the overall business transaction. In that relationship a software agent within the coordinating entity's systems plays the BTP role of Superior for a given transaction and one or more software agents within the systems of the party play the BTP role of Inferior. Each Inferior has one Superior, therefore, while a single Superior may

407	have multiple Inferiors within each party to the transaction, and may be related to Inferiors
408	within multiple parties. Each Superior:Inferior pair exchanges protocol-defined messages.
409	
410	An Inferior is associated with some set of operation invocations that creates effect
411	(provisional or tentative changes) within the party, for a given business transaction. The
412	Inferior is responsible for reporting to its related Superior whether its associated operations'
413	effect can be confirmed/cancelled. A Superior is responsible for gathering the reports of all of
414	its Inferiors, in order to ascertain which should be cancelled or confirmed. For example, if a
415	Superior is acting as an atomic Coordinator it will treat any Inferior which cannot prepare to
416	cancel/confirm as having veto power over the whole business transaction, causing the
417	Superior to instruct all its Inferiors to cancel. A Superior may, under the dictates of a
418	controlling application, increase or reduce the set of Inferiors to which a common confirm or
419	cancel outcome may be delivered. Thus, the set of prepared Inferiors may be larger than the
420	set of confirmed Inferiors.
421	
422	A Inferior:Superior relationship is typically established in relation to one or more application
423	messages sent from one part of the application (linked to the Superior) to some other part of
424	the application to request the performance of operations that are to be subject to the confirm
425	or cancel decision of the Superior. If an application is divided between a client and a service,
426	which use RPCs to communicate application requests and responses, then the client would
427	typically be associated with the Superior and the service would typically host the Inferior(s).
428	(BTP does not mandate such an application topology nor does it require the use of RPC or
429	any other application communication paradigm.)
430	
431	BTP defines a CONTEXT message that can be sent "in relation to" such application
432	messages. On receipt of a CONTEXT, one or more Inferiors are created and "enrolled" with
433	the Superior, establishing the Superior:Inferior relationships. The particular mechanisms by
434	which a CONTEXT is "related" to application is an issue for the application protocol and its
435	binding to carrier mechanisms. BTP does not require that the enrolment is requested by any
436	particular entity – in a particular implementation this may be done by the Inferior itself, by
437	parts of the application or by other entities involved in the transmission of the CONTEXT
438	and the application messages. BTP defines a CONTEXT_REPLY message that can be sent
439	on the return path of the CONTEXT to indicate whether the enrolment was successful.
440	Without CONTEXT_REPLY it would be possible for a Superior to have an incorrect view of
441	which Inferiors it was supposed to involve in its confirm decision.
442	
443	It should be noted that this BTP specification recognises that:

- 444 ❑ an Inferior may itself be a Superior to other BTP Inferiors; this occurs when some of
- 445 the operations associated with the Inferior involve other application elements whose
- 446 operations are to be subject to the confirm/cancel instruction sent to the Inferior. The
- 447 specification treats any lower Inferiors as part of the associated operations;

- 448 ❑ the requirement on an Inferior to be able to confirm or cancel does not include any
- 449 specific mechanism to determine the isolation of the effects of operations; the
- 450 requirement is only that the Inferior is able to confirm or cancel the operations, as
- 451 their effects are known to the Superior and the application directly in contact with the
- 452 Superior. Thus the confirm-or-cancel requirement may be achieved by performing all
- 453 the operations and remembering a compensating counter operation (that will be
- 454 triggered by a cancel order); or by remembering the operations (having checked they

455	are valid) and performing them only if a confirm order is received; or by forbidding
456	any other access to data changed by the operations and releasing them in their
457	unchanged state (if cancelled) or their changed state (if confirmed); or by various
458	combinations of these. In addition, a cancellation may not return data to their original
459	state, but only to a state accepted by the application as appropriate to a cancelled
460	operation.
461	
462	
463	
464	
465	
466	
467	

Part 2. Normative Specification of BTP

Actors, Roles and Relationships

Actors are software agents which process computations. BTP actors are addressable for the purposes of receiving application and BTP protocol messages transmitted over some underlying communications or carrier protocol. (See section "Addressing" for more detail.)

BTP actors play roles in the sending, receiving and processing of messages. These roles are associated with responsibilities or obligations under the terms of software contracts defined by this specification. (These contracts are stated formally in the sections entitled "Abstract Messages and Associated Contracts" and "State Tables".) A BTP actor's computations put the contracts into effect.

A role is defined and described in terms of a single business transaction. An implementation supporting a role may, as an addressable entity, play the same role in multiple business transactions, simultaneously or consecutively, or a separate addressable entity may be created for each transaction. This is a choice for the implementer, and the addressing mechanisms allow interoperation between implementations that make different choices.

Within a single transaction, one actor may play several roles, or each role may be assigned to a distinct actor. This is again a choice for the implementer. An actor playing a role is termed an "actor-in-role".

Actors may interoperate, in the sense that the roles played by actors may be implemented using software created by different vendors for each actor-in-role. The section "Conformance", gives guidelines on the groups of roles that may be implemented in a partial, interoperable implementation of BTP.

The descriptions of the roles concentrate on the normal progression of a business transaction, and some of the more important divergences from this. They do not cover all exception cases – the message set definition and the state tables provide a more comprehensive specification.

Note – A BTP role is approximately equivalent to an interface in some distributed computing mechanisms, or a port-type in WSDL. The definition of a role includes behaviour.

Relationships

There are two primary relationships in BTP.

- ❑ Between an application element that determines that a business transaction should be completed (the role of Terminator) and the BTP actor at the top of the transaction tree (the role of Decider);

510

511 ❑ Between BTP actors within the tree, where one (the Superior) will inform the other
512 (the Inferior) what the outcome decision is.
513

514 These primary relationships are involved in arriving at a decision on the outcome of a
515 business transaction, and propagating that decision to all parties to the transaction. Taking the
516 path that is followed when a business transaction is confirmed:

517 1. The Terminator determines that the business transaction should confirm, if it can; or
518 (for a Cohesion), which parts should confirm

519 2. The Terminator asks the Decider to apply the desired outcome to the tree, if it can
520 guarantee the consistency of the confirm decision

521 3. The Decider, which is Superior to one or more Inferiors, asks its Inferiors if they can
522 agree to a confirm decision (for a Cohesion, this may not be all the Inferiors)

523 4. If any of those Inferiors are also Superiors, they ask their Inferiors and so on down
524 the tree

525 5. Inferiors that are not Superiors report if they can agree to a confirm to their Superior

526 6. Inferiors that are also Superiors report their agreement only if they received such
527 agreement from their Inferiors, and can agree themselves

528 7. Eventually agreement (or not) is reported to the Decider. If all have agreed, the
529 Decider makes and persists the confirm decision (hence the term "Decider" – it
530 decides, everything else just asked); if any have disagreed, or if the confirm decision
531 cannot be persisted, a cancel decision is made

532 8. The Decider, as Superior tells its Inferiors of the outcome

533 9. Inferiors that are also Superiors tell their Inferiors, recursively down the tree

534 10. The Decider replies to the Terminator's request to confirm, reporting the outcome
535 decision
536

537 There are other relationships that are secondary to Terminator:Decider, Superior:Inferior,
538 mostly involved in the establishment of the primary relationships.
539

540 The two primary relationships are linked in that a Decider is a Superior to one or more
541 Inferiors. There are also similarities in the semantics of some of the exchanges (messages)
542 within the relationships. However they differ in that
543

544 1. All exchanges between Terminator and Decider are initiated by the Terminator (it is
545 essentially a request/response relationship); either of Superior or Inferior may initiate
546 messages to the other
547

548 2. The Superior:Inferior relationship is recoverable – depending on the progress of the
549 relationship, the two sides will re-establish their shared state after failure; the
550 Terminator:Decider relationship is not recoverable
551

552   3. The nature of the Superior:Inferior relationship requires that the two parties know of
553   each other's addresses from when the relationship is established; the Decider does not
554   need to know the address of the Terminator (provided it has some way of returning
555   the response to a received message).
556
557 In the following sections, the responsibility of each role is defined, and the messages that are
558 sent or received by that role are listed. Note that some roles exist only to have a name for an
559 actor that issues a message and receives a reply to that message. Some of these roles may be
560 played by several actors in the course of a single business transaction.
561

Roles involved in the Superior:Inferior relationship

562
563

Superior

564
565
566 Accepts enrolments from Inferiors, establishing a Superior:Inferior relationship with each. In
567 cooperation with other actors and constrained by the messages exchanged with the Inferior,
568 the Superior determines the **Outcome** applicable to the Inferior and informs the Inferior by
569 sending CONFIRM or CANCEL. This outcome can be confirm only if a PREPARED
570 message is received from the Inferior, and if a record, identifying the Inferior can be
571 persisted. (Whether this record is also a record of a confirm decision depends on the
572 Superior's position in the business transaction as a whole.). The Superior must retain this
573 persistent record until it receives a CONFIRMED (or, in exceptional cases, CANCELLED or
574 HAZARD) from the Inferior.
575
576 A Superior may delegate the taking of the confirm or cancel decision to an Inferior, if there is
577 only one Inferior, by sending CONFIRM_ONE_PHASE.
578
579 A Superior may be Atomic or Cohesive. An Atomic Superior will apply the same decision to
580 all of its Inferiors; a Cohesive Superior can apply confirm some Inferiors and cancel others,
581 or may confirm some after others have reported cancellation. The set of Inferiors that the
582 Superior confirms (or attempts to confirm) is called the "confirm-set".
583
584 If RESIGN is received from an Inferior, the Superior:Inferior relationship is ended; the
585 Inferior has no further effect on the behaviour of the Superior as a whole.
586
587 A Superior receives
588
589   ENROL
590
591 to enrol a new Inferior, establishing a new Superior:Inferior relationship.
592
593 A Superior sends
594
595   ENROLLED
596
597 in reply to ENROL, if the appropriate parameter on the ENROL asked for the reply.
598
599 A Superior sends

```
600
601            PREPARE
602            CONFIRM
603            CANCEL
604            RESIGNED
605            CONFIRM_ONE_PHASE
606            SUPERIOR_STATE
607
608    to an enrolled Inferior.
609
610    A Superior receives
611
612            PREPARED
613            CANCELLED
614            CONFIRMED
615            HAZARD
616            RESIGN
617            INFERIOR_STATE
618
619    from an enrolled Inferior.
620
```

Inferior

Responsible for applying the Outcome to some set of associated operations – the application determines which operations are the responsibility of a particular Inferior.

An Inferior is **Enrolled** with a single Superior (hereafter referred to as "its Superior"), establishing a Superior:Inferior relationship. If the Inferior is able to ensure that either a confirm or cancel decision can be applied to the associated operations, and can persist information to retain that condition, it sends a PREPARED message to the Superior. When the Outcome is received from the Superior, the Inferior applies it, deletes the persistent information, and replies with CANCELLED or CONFIRMED as appropriate.

If an Inferior is unable to come to a prepared state, it cancels the associated operations and informs the Superior with a CANCELLED message. If it is unable to either come to a prepared state, or to cancel the associated operations, it informs the Superior with a HAZARD message.

An Inferior that has become prepared may, exceptionally, make an autonomous decision, to be applied to the associated operations, without waiting for the Outcome from the Superior. It is required to persist this autonomous decision and report it to the Superior with CONFIRMED or CANCELLED as appropriate. If, when CONFIRM or CANCEL is received, the autonomous decision and the decision received from the Superior are contradictory, the Inferior must retain the record of the autonomous decision until receiving a CONTRADICTION message.

An Inferior receives

648   PREPARE
649   CONFIRM
650   CANCEL
651   RESIGNED
652   CONFIRM_ONE_PHASE
653   SUPERIOR_STATE
654
655   from its Superior.
656
657   An Inferior sends
658
659   PREPARED
660   CANCELLED
661   CONFIRMED
662   HAZARD
663   RESIGN
664   INFERIOR_STATE
665
666   to its Superior.
667
668   An Inferior receives REQUEST_STATUS and replies with STATUS. If it is also a Superior,
669   the STATUS concerns the Inferior as a whole.
670

Enroller

671

672

673   Causes the enrolment of an Inferior with a Superior. This role is distinguished because in
674   some implementations the enrolment request will be performed by the application, in some
675   the application will ask the actor that will play the role of Inferior to enrol itself, and a
676   Factory may enrol a new Inferior (which will also be Superior) as a result of receiving
677   BEGIN&CONTEXT.
678
679   An Enroller sends
680
681   ENROL
682
683   to a Superior.
684
685   An Enroller receives
686
687   ENROLLED
688
689   in reply to ENROL if the Enroller asked for a response when the ENROL was sent.
690
691   An ENROL message sent from an Enroller that did not require an ENROLLED response may
692   be modified *en route* to the Superior by an intermediate actor to ask for an ENROLLED
693   response to be sent to the intermediate. (This may occur in the "one-shot" scenario, where an
694   ENROL/no-rsp-req is received in relation to a CONTEXT_REPLY/related; the receiver of
695   the CONTEXT_REPLY will need to ensure the enrolment is successful).

696

Participant

An Inferior which is specialized for the purposes of an application. Some application operations are associated directly with the Participant, which is responsible for determining whether a prepared condition is possible for them, and for applying the outcome. ("associated directly" as opposed to involving another BTP Superior:Inferior relationship, in which this actor is the Superior).

The associated operations may be performed by the actor that has the role of Participant, or they may be performed by another actor, and only the confirm/cancel application is performed by the Participant.

In either case, the Participant, as part of becoming prepared (i.e. before it can send PREPARED to the Superior), will persist information allowing it apply a confirm decision to the operations and to apply a cancel decision. The nature of this information depends on the operations.

Note – Possible approaches are:

- o The operations may be performed completely and the Participant persists information to perform counter-effect operations (compensating operations) to apply cancellation;

- o The operations may be just checked and not performed at all; the Participant persists information to perform them to apply confirmation;

- o The Participants persists the prior state of data affected by the operations and the operations are performed; the Participant restores the prior state to apply cancellation;

- o As the previous, but other access to the affected data if forbidden until the decision is known

Sub-coordinator

An Inferior which is also an Atomic Superior.

A sub-coordinator is the Inferior in one Superior:Inferior relationship and the Superior in one or more Superior:Inferior relationships.

From the perspective of its Superior (the one the sub-coordinator is Inferior to), there is no difference between a sub-coordinator and any other Inferior. From this perspective, the "associated operations" of the sub-coordinator as an Inferior include the relationships with its Inferiors.

739 A sub-coordinator does not become prepared (and send PREPARED to its Superior) until and
740 unless it has received PREPARED (or RESIGN) from all its Inferiors. The outcome is
741 propagated to all Inferiors.
742

Sub-composer

745 An Inferior which is also a Cohesive Superior.

747 Like a sub-coordinator, a sub-composer cannot be distinguished from any other Inferior from
748 the perspective of its Superior.

750 A sub-composer is similar to a sub-coordinator, except that the constraints linking the
751 different Inferiors concern only those Inferiors in the confirm-set. How the confirm-set is
752 controlled, and when is not defined in this specification.

754 If the sub-composer is instructed to cancel, by receiving a CANCEL message from its
755 Superior, the cancellation is propagated to all its Inferiors.

Roles involved in the Terminator:Decider relationship

Decider

762 A Superior that is not the Inferior on a Superior:Inferior relationship. It is the top-node in the
763 transaction tree and receives requests from a Terminator as to the desired outcome for the
764 business transaction. If the Terminator asks the Decider to confirm the business transaction, it
765 is the responsibility the Decider to finally take the confirm decision. The taking of the
766 decision is synonymous with the persisting of information identifying the Inferiors that are to
767 be confirmed. An Inferior cannot be confirmed unless PREPARED has been received from it.

769 A Decider is instructed to cancel by receiving CANCEL/whole.

771 A Decider that is an Atomic Superior (all Inferiors will have the same outcome) is a
772 Coordinator. A Decider that is a Cohesive Superior (some Inferiors may cancel, some
773 confirm) is a Cohesion.

775 All Deciders receive
776   REQUEST_CONFIRM
777   CANCEL/whole
778   REQUEST_STATUSES

780 All Deciders send
781   CONFIRMED
782   CANCELLED
783   INFERIOR_STATUSES

785 An Decider also REQUEST_STATUS and replies with STATUS, reporting its state as a
786 whole.

Coordinator

A Decider that is an Atomic Superior. The same outcome decision will be applied to all Inferiors (excluding any from which RESIGN is received).

PREPARED must be received from all remaining Inferiors for a confirm decision to be taken.

A Coordinator must make a cancel decision if
  it is instructed to cancel by the Terminator
  if CANCELLED is received from any Inferior
  if it is unable to persist a confirm decision

Composer

A Decider that is a Cohesive Superior. If the Terminator requests confirmation of the Cohesion, that request will determine the confirm-set of the Cohesion.

PREPARED must be received from all Inferiors in the confirm-set (excluding any from which RESIGN is received) for a confirm decision to be taken.

A Composer must make a cancel decision (applying to all Inferiors) if
  it is instructed to cancel by the Terminator
  if CANCELLED is received from any Inferior in the confirm-set
  if it is unable to persist a confirm decision

A Composer may be asked to prepare some or all of its Inferiors by receiving PREPARE. It issues PREPARE to any of those Inferiors from which none of PREPARED, CANCELLED or RESIGNED have been received, and replies to the PREPARE with INFERIOR_STATUSES.

A Composer may be asked to cancel some of its Inferiors, but not itself, by receiving CANCEL/inferiors.

In addition to the messages received by the Composer as a Decider, it receives
  PREPARE
  CANCEL/inferiors

Terminator

Asks a Decider to confirm the business transaction, or instructs it to cancel all or (for a Cohesion) part of the business transaction.

All communications between Terminator and Decider are initiated by the Terminator. A Terminator is usually an application element.

A request to confirm is made by sending REQUEST_CONFIRM to the target Decider. If the Decider is a Cohesion Composer, the Terminator may select which of the Composer's

835   Inferiors are to be included in the confirm-set. If the Decider is an Atom Coordinator, all
836   Inferiors are included. After applying the decision, the Decider replies with CONFIRMED,
837   CANCELLED or (in the case of problems) INFERIOR_STATUSES.
838
839   A Terminator may ask a Composer (but not a Coordinator) to prepare some or all of its
840   Inferiors with PREPARE/inferiors. The Composer replies with INFERIOR_STATUSES.
841
842   A Terminator may send CANCEL to instruct the Decider to cancel the whole business
843   transaction, or, if it is a Cohesion Composer, some of its Inferiors. The Decider replies with
844   CANCELLED, or for a selective cancel or in the case of problems, INFERIOR_STATUSES.
845
846   A Terminator may check the status of the Inferiors of the Decider by sending
847   REQUEST_STATUSES. The Decider replies with INFERIOR_STATUSES.
848
849   A Terminator sends
850   REQUEST_CONFIRM
851   CANCEL
852   PREPARE/inferiors
853   REQUEST_STATUSES
854
855   A Terminator receives
856   CONFIRMED
857   CANCELLED
858   INFERIOR_STATUSES
859

Initiator

861
862   Requests a **Factory** to create a Superior – this will either be a Decider (representing a new
863   top-level business transaction) or a sub-coordinator or sub-composer to be the Inferior of an
864   existing business transaction.
865
866   An Initiator sends
867
868   BEGIN
869   BEGIN & CONTEXT
870
871   to a Factory, and receives in reply
872
873   BEGUN & CONTEXT
874

Factory

876
877   Creates Superiors and returns the CONTEXT for the new Superior. The following types of
878   Superior are created :
879
880   Decider, which may either
881   Composer or
882   Coordinator

| 883 | Sub-composer |
| 884 | Sub-coordinator |

886   A Factory receives

888   BEGIN
889   BEGIN & CONTEXT

891   and replies with

893   BEGUN & CONTEXT

895   If the BEGIN has no related CONTEXT, the Factory creates a Decider, either a Cohesion
896   Composer or an Atom Coordinator, as determined by the "superior type" parameter on the
897   BEGIN.

899   If the BEGIN has a related CONTEXT, the new Superior is also enrolled as an Inferior of the
900   Superior identified by the CONTEXT. The new Superior is thus a sub-composer or sub-
901   coordinator, as determined by the "superior type" parameter on the BEGIN.

905   **Other roles**

907   Redirector

909   Sends a REDIRECT message to inform any actor that an address previously supplied for
910   some other actor is no longer appropriate, and to supply a new address to replace the old one.

912   A Redirector may send a REDIRECT message in response to receiving a message using the
913   old address, or may send REDIRECT at its own initiative.
914   If a Superior moves from the superior-address in its CONTEXT, or an Inferior moves from
915   the inferior-address in the ENROL message, the implementation **must** ensure that a
916   Redirector catches any inbound messages using the old address and replies with a
917   REDIRECT message giving the new address. (Note that the inbound message may itself be a
918   REDIRECT message.)

920   A Redirector **may** also be used to change the address of other BTP actors.

922   After receiving a REDIRECT message, the BTP actor **must** use the new address not the old
923   one, unless failure prevents it updating its information.

925   Status Requestor

927   Requests and receives the current status of an Inferior or a Decider. The role of Status
928   Requestor has no responsibilities – it is just a name for where the REQUEST_STATUS
929   comes from.
930

931   A Status Requestor sends

932

933   REQUEST_STATUS

934

935   and receives

936

937   STATUS

938

939   in response.

940

941   The information returned will always relate to the actor concerned in its role as an Inferior,
942   even if it also a Superior.

943

Abstract Messages and Associated Contracts

945

946   BT Protocol Messages are defined in this section in terms of the abstract information that has
947   to be communicated. These abstract messages will be mapped to concrete messages
948   communicated by a particular carrier protocol (there can be several such mappings defined).

949

950   The abstract message set and the associated state table assume the carrier protocol will

951

952   ❑   deliver messages completely and correctly, or not at all (corrupted messages will
953   not be delivered);

954

955   ❑   report some communication failures, but will not necessarily report all (i.e. not all
956   message deliveries are positively acknowledged within the carrier);

957

958   ❑   sometimes deliver successive messages in a different order than they were sent;

959

960   and

961

962   ❑   does not have built-in mechanisms to link a request and a response

963

964   Note that these assumptions would be met by a mapping to SMTP and more than met by
965   mappings to SOAP.

966

967   However, when the abstract message set is mapped to a carrier protocol that provides a richer
968   service (e.g. reports all delivery failures, guarantees ordered delivery or offers a
969   request/response mechanism), the mapping can take advantage of these features. Typically in
970   such cases, some of the parameters of an abstract message will be implicit in the carrier
971   mechanisms, while the values of other parameters will be directly represented in transmitted
972   elements.

973

974

975   **Addresses**

976

977 All of the messages except CONTEXT and CONTEXT_REPLY have a "target address"
978 parameter and many also have other address parameters. These latter identify the desired
979 target of other messages in the set. In all cases, the exact value will invariably have been
980 originally determined by the implementation that is the target or desired future target.
981
982 The detailed format of the address will depend on the particular carrier protocol, but at this
983 abstract level is considered to have three parts. The first part, the "binding name", identifies
984 the binding to a particular carrier protocol – some bindings are specified in this document,
985 others can be specified elsewhere. The second part of the address, the "binding address", is
986 meaningful to the carrier protocol itself, which will use it for the communication (i.e. it will
987 permit a message to be delivered to a receiver). The third part, "additional information", is
988 not used or understood by the carrier protocol. The "additional information" may be a
989 structured value.
990
991 When a message is actually transmitted, the "binding name" of the target address will identify
992 which carrier protocol is in use and the "binding address" will identify the destination, as
993 known to the carrier protocol. The entire binding address is considered to be "consumed" by
994 the carrier protocol implementation. All of it may may be used by the sending
995 implementation, or some of it may be transmitted in headers, or as part of a URL in the
996 carrier protocol, but then used or consumed by the receiving implementation of the carrier
997 protocol to direct the BTP message to a BTP-aware entity (BTP-aware in that it is capable of
998 interpreting the BTP messages). The "additional information" of the target address will be
999 part of the BTP message itself and used in some way by the receiving BTP-aware entity (it
1000 could be used to route the message on to some other BTP entity). Thus, for the target address,
1001 only the "additional information" field is transmitted in the BTP message and the "additional
1002 information" is opaque to parties other than the recipient.
1003
1004 For other addresses in BTP messages, all three components will be within the message.
1005
1006 All messages that concern a particular Superior:Inferior relationship have an identifier
1007 parameter for the target side as well as the compound target address. This allows full
1008 flexibility for implementation choices – an implementation can:
1009
1010   a) Use the same binding address and additional information for multiple business
1011   transactions, using the identifier parameter to locate the relevant state
1012   information;
1013   b) Use the same binding address for multiple business transactions and use the
1014   additional information to locate the information; or
1015   c) Use a different binding address for each business transaction.
1016
1017 Which of these choices is used is opaque to the entity sending the message – both parts of the
1018 address and the identifier originated at the recipient of this message (and were transmitted as
1019 parameters of earlier messages in the opposite direction). In cases b) and c), the identifier is to
1020 some extent redundant, although interoperation requires that it always be present.
1021
1022 BTP recovery requires that the state information for a Superior or Inferior is accessible after
1023 failure and that the peer can distinguish between temporary inaccessibility and the permanent
1024 non-existence of the state information. As is explained in "Redirection" below, BTP provides

1025	mechanisms – having a set of BTP addresses for some parameters, and the REDIRECT
1026	message – that make this possible, even if the recovered state information is on a different
1027	address to the original one (as ma y be the case if case c) above is used).
1028	
1029	

Request/response pairs

1031	
1032	Many of the messages combine in pairs as a request and its response. However, in some cases
1033	the response message is sent without a triggering request, or as a possible response to more
1034	than one type of request. To allow for this, the abstract message set treats each message as
1035	standalone; but where a request does expect a reply, a "reply-address" parameter will be
1036	present. For any message with a reply address parameter, in the case of certain errors, a
1037	FAULT message will be sent to the reply address instead of the expected reply.
1038	
1039	For messages which are specified as sent between Superior and Inferior, a FAULT message is
1040	sent to the peer.
1041	

Compounding messages

1043	
1044	BTP messages may be sent in combination with each other, or with other (application)
1045	messages. There are two cases:
1046	
1047	a) Sending the messages together has semantic significance. One message is said
1048	to be "related to" the other.
1049	b) Sending of the messages has no semantic significance, but is merely a
1050	convenience or optimisation. This is termed "bundling".
1051	
1052	The form A&B is used to refer to a combination where message B is sent in relation to A
1053	("relation" is asymmetric). The form A+B is used to refer to A and B bundled together.
1054	
1055	In both cases the messages will have the same binding address, but may have different
1056	"additional information" values. Unless constrained by the binding, any messages that are to
1057	be sent to the same binding address may be bundled – the fact that the binding addresses are
1058	the same is a necessary and sufficient condition for the sender to determine that the messages
1059	can be bundled.
1060	
1061	A particular and important case of related messages is where a BTP CONTEXT message is
1062	sent related to an application message. In this case, the target of the application message
1063	defines the destination of the CONTEXT message. The receiving implementation may in fact
1064	remove the CONTEXT before delivering the application message to the application (Service)
1065	proper, but from the perspective of the sender, the two are sent to the same place.
1066	The compounding mechanisms, and the multi-part address structures, support the "one-wire"
1067	and "one-shot" communication patterns.
1068	
1069	In "one-wire", all message exchanges between two sides of a Superior:Inferior relationship,
1070	including the associated application messages, pass via the same "endpoints". These
1071	"endpoints" may in fact be relays, routing messages on to particular actors within their
1072	domain. The onward routing will require some further addressing, but this has to be opaque to

1073 the sender. This can be achieved if the relaying endpoint ensures that all addresses for actors
1074 in its domain have the relays address as their binding address, and any routing information it
1075 will need in its own domain is placed in the additional information. (This may involve the
1076 relay changing addresses in messages as they pass through it on the way out). On receiving a
1077 message, it determines the within-domain destination from the received additional
1078 information (which is thus rewritten) and forwards the message appropriately. The sender is
1079 unaware of this, and merely sees addresses with the same binding address, which it is
1080 permitted to bundle. The content of the "additional information" is matter only for the relay –
1081 it could put an entire BTP address in there, or other implementation-defined information.
1082 Note that a quite different one-wire implementation can be constructed where there is no
1083 relaying, but the receiving entity effectively performs all roles, using the received identifiers
1084 to locate the appropriate state.
1085
1086 "One-shot" communication concerns the bundling of application messages, especially where
1087 the application uses a request/response paradigm. The application request is sent with a
1088 related CONTEXT message. The application response is sent with a related
1089 CONTEXT_REPLY/related, with an ENROL/no-rsp-req message and a bundled
1090 PREPARED message (assuming the operations succeeded and the Inferior has decided to be
1091 prepared). The target address of the ENROL and PREPARED (the Superior address) must
1092 have a binding address that is the same as the target address of the application response (i.e.
1093 the reply address for the client, as perceived by the Service) – otherwise the Service cannot
1094 determine that is should bundle the messages together. One-shot is thus a specialisation of
1095 one-wire.
1096
1097 With "one-shot", if there are multiple Inferiors created as a result of a single application
1098 message, there is an ENROL and PREPARED message for each sent with the application
1099 response and the CONTEXT_REPLY. If an operation fails, a CANCELLED message can be
1100 sent with the response instead of a PREPARED. If subsequent messages to the same Service,
1101 with the same related CONTEXT, have their associated operations put under the control of
1102 the same Inferior, only a CONTEXT_REPLY/completed is sent back with the response (if the
1103 new operations fail, it will be necessary to send back CONTEXT_REPLY/repudiated, or send
1104 CANCELLED).
1105

1106 Where does that last bit on one-shot, one-wire belong. It needs to be in somewhere.
1107 prf

1108

1109 Extensibility

1110
1111 To simplify interoperation between implementations of this edition of BTP with
1112 implementations of future editions, the "must-be-understood" sub-parameter as specified for
1113 Qualifiers may be defined for use with any parameter added to an existing message in a future
1114 revision of this specification. The default for "must-be-understood" shall be "true", so an
1115 implementation receiving an unrecognised parameter without a "false" value for "must-be-
1116 understood" shall not accept it (the FAULT value "UnrecognisedParameter" is available, but
1117 other errors, including lower-layer parsing/unmarshalling errors may be reported instead). If
1118 "must-be-understood" with the value "false" is present as a sub-parameter of a parameter in
1119 any message, a receiving implementation **should** ignore the parameter.
1120

1121 How the sub-parameter is associated with the new parameter is determined by the particular
1122 binding.
1123
1124 No special mechanism is provided to allow for the introduction of completely new messages.
1125

1126 **Inferior handle**
1127
1128 Some of the messages exchanged between a Terminator and a Decider are concerned with the
1129 individual Inferiors enrolled with the Decider, and not with the business transaction as a
1130 whole. These messages distinguish the Inferiors of Decider using an "inferior handle". This is
1131 created by the Decider and is unambiguous within the scope of the Decider .
1132
1133 The "inferior handle" is distinct from the "inferior identifier" passed on an ENROL message
1134 (among other places). The latter is created by the Inferior (or its enroller) and is required to be
1135 unambiguous within the scope of the address-as-inferior on the ENROL. (and unambiguous
1136 within **any** of the individual addresses in that set of BTP address - the identifier must identify
1137 the Inferior across all the places it might migrate to or that have recovery responsibility for
1138 it).
1139
1140 The "inferior handle" is only used by the Terminator to refer to the inferiors of the Decider.
1141 In messages between the Decider and its Inferiors, the address-as-inferior and inferior
1142 identifie r are used.
1143

1144 **Messages**
1145

1146 **Qualifiers**
1147
1148 All messages have a Qualifiers parameter which contains zero or more Qualifier values. A
1149 Qualifier has sub-parameters:
1150

Sub-parameter	Type
qualifier name	string
qualifier group	URI
must-be-understood	Boolean
to-be-propagated	Boolean
content	Arbitrary – depends on type

1151
1152 **Qualifier group** ensures the Qualifier name is unambiguous. Qualifiers in the
1153 same group need not have any functional relationship. The qualifier group will
1154 typically be used to identify the specification that defines the qualifier's meaning
1155 and use. Qualifiers may be defined in this or other standard specifications, in
1156 specifications of a particular community of users or of implementations or by
1157 bilateral agreement.
1158

1159	**Qualifier name** this identifies the meaning and use of the Qualifier, using a name
1160	that is unambiguous within the scope of the Qualifier group.
1161	
1162	**Must-be-understood** if this has the value "true" and the receiving entity does
1163	not recognise the Qualifier type (or does not implement the necessary
1164	functionality), a FAULT "UnsupportedQualifier" shall be returned and the
1165	message shall not be processed. Default is "true".
1166	
1167	**To-be-propagated** if this has the value "true" and the receiving entity passes the
1168	BTP message (which may be a CONTEXT, but can be other messages) onwards
1169	to other entities, the same Qualifier value shall be included. If the value is
1170	"false", the Qualifier shall not be automatically included if the BTP message is
1171	passed onwards. (If the receiving entity does support the qualifier type, it is
1172	possible a propagated message may contain another instance of the same type,
1173	even with the same Content – this is not considered propagation of the original
1174	qualifier.). Default is "false".
1175	
1176	**Content** the type (which may be structured) and meaning of the content is
1177	defined by the specification of the Qualifier.
1178	
1179	
1180	**CONTEXT**
1181	
1182	A CONTEXT is supplied by (or on behalf of) a Superior and related to one or more
1183	application messages. (The means by which this relationship is represented is determined by
1184	the binding and the binding mechanisms of the application protocol. The "superior type"
1185	parameter identifies whether the Superior will apply the same decision to all Inferiors
1186	enrolled with using the same superior identifier ("superior type" is "atom") or may apply
1187	different decisions ("superior type" is "cohesion").
1188	

Parameter	Type
address-as-superior	Set of BTP addresses
superior identifier	Identifier
superior type	cohesion/atom
qualifiers	List of qualifiers

1189	
1190	
1191	**address-as-superior** the address to which ENROL and other messages from an
1192	enrolled Inferior are to be sent. This can be a set of alternative addresses.
1193	
1194	**superior identifier** identifies the Superior within the scope of the address-as-
1195	superior
1196	
1197	**superior type** identifies whether the CONTEXT refers to a Cohesion or an
1198	Atom. Default is atom.

1199
1200
1201 **qualifiers** standardised or other qualifiers. The standard qualifier "Transaction
1202 timelimit" is carried by CONTEXT.
1203
1204 There is no target address parameter for CONTEXT as it is only transmitted in relation to the
1205 application messages.
1206
1207 The forms CONTEXT/cohesion and CONTEXT/atom refer to CONTEXT messages with the
1208 superior type with the appropriate value.
1209
1210
1211 **CONTEXT_REPLY**
1212
1213 CONTEXT_REPLY is sent after receipt of CONTEXT (related to application message(s)) to
1214 indicate whether all necessary enrolments have already completed (ENROLLED has been
1215 received) or will be completed by ENROL messages sent in relation to the
1216 CONTEXT_REPLY or if an enrolment attempt has failed. CONTEXT_REPLY may be sent
1217 related to an application message (typically the response to the application message related to
1218 the CONTEXT). In some bindings the CONTEXT_REPLY may be implicit in the application
1219 message.
1220

Parameter	Type
superior-address	BTP address
superior identifier	Identifier
completion_status	complete/related/repudiated
qualifiers	List of qualifiers

1221
1222 **superior-address** one of the addresses from the address-as-superior from the
1223 CONTEXT. (The parameter is present in CONTEXT_REPLY to disambiguate
1224 the superior identifier.)
1225
1226 **superior identifier** the superior identifier from the CONTEXT
1227
1228 **completion_status**: reports whether all enrol operations made necessary by the
1229 receipt of the earlier CONTEXT message have completed. Values are
1230

value	meaning
completed	All enrolments (if any) have succeeded already
related	At least some enrolments are to be performed by ENROL messages related to the CONTEXT_REPLY. All other enrolments (if any) have succeeded already.
repudiated	At least one enrolment has failed. The implications of receiving the CONTEXT have **not** been honoured.

1231
1232   **qualifiers** standardised or other qualifiers.
1233
1234 The form CONTEXT_REPLY/completed, CONTEXT_REPLY/related and
1235 CONTEXT_REPLY/repudiated refer to CONTEXT_REPLY messages with status having the
1236 appropriate value. The form CONTEXT_REPLY/ok refers to either of
1237 CONTEXT_REPLY/completed or CONTEXT_REPLY/related.
1238
1239 If there are no necessary enrolments (e.g. the application messages related to the received
1240 CONTEXT did not require the enrolment of any Inferiors), then
1241 CONTEXT_REPLY/completed is used.
1242
1243 If a CONTEXT_REPLY/repudiated is received, the receiving implementation **must** ensure
1244 that the business transaction will not be confirmed.
1245
1246

1247 **BEGIN**
1248
1249 A request to a Factory to create a new Business Transaction. This may either be a new top-
1250 level transaction, in which case the Composer or Coordinator will be the Decider, or the new
1251 Business Transaction may be immediately made the Inferior within an existing Business
1252 Transaction (thus creating a sub-Composer or sub-Coordinator).
1253

Parameter	Type
target address	BTP address
reply address	BTP address
transaction type	cohesion/atom
qualifiers	List of qualifiers

1254
1255   **target address** the address of the entity to which the BEGIN is sent. How this
1256   address is acquired and the nature of the entity are outside the scope of this
1257   specification.
1258
1259   **reply address** the address to which the replying BEGUN and related
1260   CONTEXT message should be sent.
1261
1262   **transaction type** identifies whether a new Cohesion or new Atom is to be
1263   created; this value will be the "superior type" in the new CONTEXT
1264
1265   **qualifiers** standardised or other qualifiers. The standard qualifier "Transaction
1266   timelimit" may be present on BEGIN, to set the timelimit for the new business
1267   transaction and will be copied to the new CONTEXT. The standard qualifier
1268   "Inferior name" may be present if there is a CONTEXT related to the BEGIN.
1269

	1270	A new top-level Business Transaction is created if there is no CONTEXT related to the
1271	BEGIN. A Business Transaction that is to be Inferior in an existing Business Transaction is
1272	created if the CONTEXT message for the existing Business Transaction is related to the
1273	BEGIN. In this case, the Factory is responsible for enrolling the new Composer or
1274	Coordinator as an Inferior of the Superior identified in that CONTEXT.
1275

1276   Note – This specification does not provide a standardised means to
1277   determine which of the Inferiors of a sub-Composer are in its confirm set.
1278   This is considered part of the application:inferior relationship.

1279

1280   The forms BEGIN/cohesion and BEGIN/atom refer to BEGIN with "transaction type" having
1281   the corresponding value.

1282

1283   Types of FAULT possible (sent to Reply address)

1284

1285   **General**

1286

1287   **BEGUN**

1288

1289   BEGUN is a reply to BEGIN. There is always a related CONTEXT, which is the CONTEXT
1290   for the new business transaction.

1291

Parameter	Type
target address	BTP address
address-as-decider	Set of BTP address
transaction-identifier	Identifier
inferior-handle	Handle
address-as-inferior	Set of BTP address
qualifiers	List of qualifiers

1292

1293   **target address** the address to which the BEGUN is sent. This will be the reply
1294   address from the BEGIN.

1295

1296   **address-as-decider** for a top-level transaction (no CONTEXT related to the
1297   BEGIN), this is the address to which PREPARE, REQUEST_CONFIRM,
1298   CANCEL and REQUEST_STATUS messages are to be sent; if a CONTEXT
1299   was related to the BEGIN this parameter is absent

1300

1301   **transaction-identifier** identifies the new Composer or Coordinator within the
1302   scope of the address-as-decider. If this is not a top-level transaction, the
1303   transaction-identifier is optional, but if present shall be the inferior-identifier used
1304   in the enrolment with the Superior identified by the CONTEXT related to the
1305   BEGIN.

1306

1307   **inferior handle** Shall be absent if this is a top-level transaction and may or may
1308   not be present otherwise. (Presence or absence will be determined by the nature
1309   of the Superior identified in the CONTEXT related to the BEGIN). If present, the
1310   inferior handle will identify this new business transaction as in the inferiors-list
1311   parameters in messages between the Superior identified in the CONTEXT related
1312   to the BEGIN (acting as a Decider) and its Terminator. The value shall be
1313   different for each enrolled Inferior of that Superior.

1314

1315   **address-as-inferior** This parameter shall be absent if this is a top-level
1316   transaction and may be present, at implementation option otherwise. If present, it
1317   shall be the address-as-inferior used in the enrolment with the Superior identified
1318   by the CONTEXT related to the BEGIN. If this is a top-level transaction

1319

1320   **qualifiers** standardised or other qualifiers.

1321

1322   At implementation option, the "address-as-decider" and/or "address-as-inferior" and the
1323   "address-as-superior" in the related CONTEXT may be the same or may be different. There
1324   is no general requirement that they even use the same bindings. Any may also be the same as
1325   the target address of the BEGIN message (the inferior identifier on messages will ensure they
1326   are applied to the appropriate Composer or Coordinator).

1327

1328   No FAULT messages are issued on receiving BEGUN.

1329

1330   **ENROL**

1331

1332   A request to a Superior to ENROL an Inferior. This is typically issued after receipt of a
1333   CONTEXT message in relation to an application request.
1334   The actor issuing ENROL plays the role of Enroller.

1335

Parameter	type
target address	BTP address
superior identifier	Identifier
reply requested	Boolean
reply address	BTP address
address-as-inferior	Set of BTP address
inferior identifier	Identifier
Qualifiers	List of qualifiers

1336

1337   **target address** the address to which the ENROL is sent. This will be the
1338   address-as-superior from the CONTEXT message.

1339

1340   **superior identifier**. The superior identifier as on the CONTEXT message

1342   **reply requested** true if an ENROLLED response is required, false otherwise.
1343   Default is false.

1344

1345   **reply address** the address to which a replying ENROLLED is to be sent, if
1346   "reply requested" is true. If this field is absent and "reply requested" is true, the
1347   ENROLLED should be sent to the "address-as-inferior" (or one of them, at
1348   sender's option)

1349

1350   **address-as-inferior** the address to which PREPARE, CONFIRM, CANCEL and
1351   SUPERIOR_STATE messages for this Inferior are to be sent.

1352

1353   **inferior identifier** an identifier that unambiguously identifies this Inferior within
1354   the scope of any of the address-as-inferior set of BTP-addresses.

1355

1356   **qualifiers** standardised or other qualifiers. The standard qualifier "Inferior
1357   name" may be present.

1358

1359   Types of FAULT possible (sent to Reply address)

1360

1361   *General*
1362   *InvalidSuperior* – if superior identifier is unknown
1363   *DuplicateInferior* – if inferior with at least one of the set address-as-
1364   inferior the same and the same inferior identifier is already enrolled
1365   *WrongState* – if it is too late to enrol new Inferiors (generally if the
1366   Superior has already sent a PREPARED message to its superior or
1367   terminator, or if it has already issued CONFIRM to other Inferiors).

1368

1369   The form ENROL/rsp-req refers to an ENROL message with "reply requested" having the
1370   value "true"; ENROL/no-rsp-req refers to an ENROL message with "reply requested" having
1371   the value "false"

1372

1373   ENROL/no-rsp-req is typically sent in relation to CONTEXT_REPLY/related. ENROL/rsp-
1374   req is typically when CONTEXT_REPLY/completed will be used (after the ENROLLED
1375   message has been received.)

1376

1377 **ENROLLED**

1378

1379   Sent from Superior in reply to an ENROL/rsp-req message, to indicate the Inferior has been
1380   successfully enrolled (and will therefore be included in the termination exchanges)

1381

Parameter	Type
target address	BTP address
inferior identifier	Identifier
inferior-handle	Handle

| | Qualifiers | List of qualifiers |

1382

1383 **target address** the address to which the ENROLLED is sent. This will be the
1384 reply address from the ENROL message (or one of the address-as-inferiors if the
1385 reply address was empty)

1386

1387 **inferior identifier** The inferior identifier as on the ENROL message

1388

1389 **inferior handle** the inferior handle that will identify this newly enrolled Inferior
1390 in the inferiors-list parameters in messages between the Superior (acting as a
1391 Decider) and its Terminator. This parameter is optional. The value shall be
1392 different for each enrolled Inferior of the Superior.

1393

1394 **qualifiers** standardised or other qualifiers.

1395

1396 No FAULT messages are issued on receiving ENROLLED.

1397
1398

1399 RESIGN

1400

1401 Sent from an enrolled Inferior to the Superior to remove the Inferior from the enrolment. This
1402 can only be sent if the operations of the business transaction have had no effect as perceived
1403 by the Inferior.

1404

1405 RESIGN may be sent in response to a PREPARE message (instead of a PREPARED), or at
1406 any point prior to the sending of a PREPARED or CANCELLED message.

1407

Parameter	type
target address	BTP address
superior identifier	identifier
address-as-inferior	Set of BTP address
inferior identifier	identifier
response requested	Boolean
Qualifiers	List of qualifiers

1408

1409 **target address** the address to which the RESIGN is sent. This will be the
1410 superior address as used on the ENROL message.

1411

1412 **superior-identifier** The superior identifier as on the ENROL message

1413

1414 **address-as-inferior** The address-as-inferior as on the earlier ENROL message
1415 (with the inferior identifier, this determines who the message is from)

1416

1417 **inferior-identifier** The inferior identifier as on the earlier ENROL message

1418
1419   **response-requested** is set to "true" if a RESIGNED response is required.
1420
1421   **qualifiers** standardised or other qualifiers.
1422
1423 Note -- RESIGN is equivalent to readonly vote in some other protocols, but can be issued
1424 early.
1425
1426 Types of FAULT possible (sent to address-as-inferior)
1427
1428   *General*
1429   *InvalidSuperior* – if superior identifier is unknown
1430   *InvalidInferior* – if no ENROL had been received for this address-as-
1431   inferior and identifier (Inferior Identity)
1432   *WrongState* – if a PREPARED or CANCELLED has already been
1433   received by the Superior from this Inferior
1434
1435 The form RESIGN/rsp-req refers to an RESIGN message with "reply requested" having the
1436 value "true"; RESIGN /no-rsp-req refers to an RESIGN message with "reply requested"
1437 having the value "false"
1438
1439
1440 **RESIGNED**
1441
1442 Sent in reply to a RESIGN/rsp-req message.
1443

Parameter	Type
target address	BTP address
inferior identifier	Identifier
qualifiers	List of qualifiers

1444
1445   **target address** the address to which the RESIGNED is sent. This will be the
1446   address-as-inferior from the ENROL message.
1447
1448   **inferior identifier** The inferior identifier as on the earlier ENROL message for
1449   this Inferior.
1450
1451   **qualifiers** standardised or other qualifiers.
1452
1453 After receiving this message the Inferior will not receive any more messages with this
1454 address-as-inferior and identifier.
1455
1456 No FAULT messages are issued on receiving RESIGNED.
1457
1458 **PREPARE**
1459

1460	Sent from Superior to an Inferior from whom ENROL but neither CANCELLED nor
1461	RESIGN have been received, requesting a PREPARED message. PREPARE can be sent after
1462	receiving a PREPARED message.
1463	
1464	Sent from a Terminator to a Composer to tell it to prepare all or some of its inferiors, by
1465	sending PREPARE to any that have not already sent PREPARED, RESIGN or
1466	CANCELLED to the Composer. If the inferiors-list parameter is absent, the request applies to
1467	all the inferiors; if the parameter is present, it applies only to the identified inferiors of the
1468	Composer.
1469	

Parameter	Type
target address	BTP address
inferior identifier	Identifier
reply address	BTP address
transaction-identifier	Identifier
inferiors-list	List of inferior handles
qualifiers	List of qualifiers

1470

1471 **target address** the address to which the PREPARE message is sent. When sent
1472 from Superior to Inferior, this will be the address-as-inferior from the ENROL
1473 message,. When sent from Terminator to Composer, this will be the decider-
1474 address from the BEGUN message .

1475

1476 **inferior identifier** When sent from Superior to Inferior, the inferior identifier as
1477 on the earlier ENROL message. This parameter shall be absent when sent from
1478 Terminator to Composer.

1479

1480 **reply address** When sent from Terminator to Composer, the address of the
1481 Terminator sending the PREPARE message. This parameter shall be absent when
1482 sent from Superior to Inferior.

1483

1484 **transaction identifier** When sent from Terminator to Composer, identifies the
1485 Composer and will be the transaction-identifier from the BEGUN message.. This
1486 parameter shall be absent when sent from Superior to Inferior.

1487

1488

1489 **inferiors-list** When sent from Terminator to Composer, defines which of the
1490 Inferiors of this Composer preparation is requested for. If this parameter is absent
1491 when sent to a Composer, the PREPARE applies to all Inferiors. This parameter
1492 shall be absent when sent from Superior to Inferior.

1493

1494

1495 **qualifiers** standardised or other qualifiers. The standa rd qualifier "Minimal
1496 inferior timeout" is carried by PREPARE.

1497

1498
1499 On receiving PREPARE, an Inferior **should** reply with a PREPARED, CANCELLED or
1500 RESIGN.
1501
1502 When sent to a Composer, for all Inferiors identified in the inferiors-list parameter (all
1503 Inferiors if the parameter is absent), from which none of PREPARED, CANCELLED or
1504 RESIGNED has been received, the Composer shall issue PREPARE. It will reply to the
1505 Terminator, using the reply address on the PREPARE message, sending an
1506 INFERIOR_STATUSES message giving the status of the Inferiors identified on the inferiors-
1507 list parameter (all of them if the parameter was absent).
1508
1509 Types of FAULT possible (sent to Superior address)
1510
1511 *General*
1512 *UnknownTransaction* – if the transaction-identifier is unknown
1513 *InvalidInferior* – if inferior identifier is unknown, or an inferior-handle
1514 on the inferiors-list is unknown
1515 *WrongState* – if a CONFIRM or CANCEL has already been received by
1516 this Inferior; if a REQUEST_CONFIRM or CANCEL/whole has already
1517 been received by this Composer.
1518
1519 The form PREPARE/whole refers to a PREPARE message sent to a Composer where the
1520 "inferiors-list" parameter is absent. The form PREPARE/inferiors refers to a PREPARE
1521 message sent to a Composer where the "inferiors-list" parameter is present. The unqualified
1522 form PREPARE is used for a PREPARE message sent to an Inferior.
1523
1524 ## PREPARED
1525
1526 Sent from Inferior to Superior, either unsolicited or in response to PREPARE, but only when
1527 the Inferior has determined the operations associated with the Inferior can be confirmed and
1528 can be cancelled, as may be instructed by the Superior. The level of isolation is a local matter
1529 (i.e. is the Inferiors choice, as constrained by the shared understanding of the application
1530 exchanges) – other access may be blocked, may see applied results of operation or may see
1531 original state.
1532

Parameter	Type
target address	BTP address
superior identifier	Identifier
address-as-inferior	Set of BTP address
inferior identifier	Identifier
default is cancel	Boolean
qualifiers	List of qualifiers

1533

1534	**target address** the address to which the PREPARED is sent. This will be the
1535	Superior address as on the ENROL message.
1536	
1537	**superior identifier** When the message is sent from an Inferior to the Superior,
1538	the superior identifier as on the ENROL message
1539	
1540	**address-as-inferior** When the message is sent from an Inferior to the Superior,
1541	the address-as-inferior as on the earlier ENROL message (with the inferior
1542	identifier, this determines who the message is from)
1543	
1544	**inferior identifier** The inferior identifier as on the ENROL message
1545	
1546	**default is cancel** if "true", the Inferior states that if the outcome at the Superior
1547	is to cancel the operations associated with this Inferior, no further messages need
1548	be sent to the Inferior. If the Inferior does not receive a CONFIRM message, it
1549	will cancel the associated operations. The value "true" will invariably be used
1550	with a qualifier indicating under what circumstances (usually a timeout) an
1551	autonomous decision to cancel will be made. If "false", the Inferior will expect
1552	a CONFIRM or CANCEL message as appropriate, even if qualifiers indicate that
1553	an autonomous decision will be made.
1554	
1555	**qualifiers** standardised or other qualifiers. The standard qualifier "Inferior
1556	timeout" may be carried by PREPARED.

1557
On sending a PREPARED, the Inferior undertakes to maintain its ability to confirm or cancel
the effects of the associated operations until it receives a CONFIRM or CANCEL message.
Qualifiers may define a time limit or other constraints on this promise. The "default is
cancel" parameter affects only the subsequent message exchanges and does not of itself state
that cancellation will occur.

1564
Types of FAULT possible (sent to address-as-inferior)

1566
General
InvalidSuperior – if Superior identifier is unknown
InvalidInferior – if no ENROL has been received for this address-as-
inferior and identifier, or if RESIGN has been received from this Inferior

1571
The form PREPARED/cancel refers to a PREPARED message with "default is cancel" =
"true". The unqualified form PREPARED refers to a PREPARED message with "default is
cancel" = "false".

CONFIRM

1578
Sent by the Superior to a Inferior from whom PREPARED has been received.

1579
| Parameter | Type |

target address	BTP address
inferior identifier	Identifier
qualifiers	List of qualifiers

1580

1581   **target address** the address to which the CONFIRM message is sent. This will
1582   be the address-as-inferior from the ENROL message.

1583

1584   **inferior identifier** The inferior identifier as on the earlier ENROL message for
1585   this Inferior.

1586

1587   **qualifiers** standardised or other qualifiers.

1588

1589 On receiving CONFIRM, the Inferior is released from its promise to be able to undo the
1590 operations of associated with the Inferior. The effects of the operations can be made available
1591 to everyone (if they weren't already)

1592

1593 Types of FAULT possible (sent to Superior address)

1594

1595   *General*
1596   *InvalidInferior* – if inferior identifier is unknown
1597   *WrongState* – if no PREPARED has been sent by, or if CANCEL has
1598   been received by this Inferior.

1599
1600

1601 **CONFIRMED**

1602

1603 Sent after the Inferior has applied the confirmation, both in reply to CONFIRM or when the
1604 Inferior has made an autonomous confirm decision, and in reply to a
1605 CONFIRM_ONE_PHASE if the Inferior decides to confirm its associated operations.

1606

1607 CONFIRMED is also sent by Decider to a Terminator in reply to REQUEST_CONFIRM if
1608 all of the confirm-set confirms (and, for a Cohesion, all other Inferiors cancel) without
1609 reporting hazards.

1610

Parameter	Type
target address	BTP address
superior identifier	Identifier
address-as-inferior	Set of BTP address
inferior identifier	Identifier
address-as-decider	BTP address
transaction-identifier	identifier
confirm received	Boolean

	qualifiers	List of qualifiers

target address the address to which the CONFIRMED is sent. When sent by an Inferior to a Superior, this will be the Superior address as on the CONTEXT message. When sent from a Decider to a Terminator it will be the reply address from the REQUEST_CONFIRM message.

superior identifier When the message is sent from an Inferior to the Superior, this shall be the superior identifier as on the CONTEXT message. This parameter shall be absent when CONFIRMED is sent from Decider to Terminator.

address-as-inferior When the message is sent from an Inferior to the Superior, this shall be the address-as-inferior as on the earlier ENROL message (with the inferior identifier, this determines who the message is from). This parameter shall be absent when CONFIRMED is sent from Decider to Terminator.

inferior identifier When the message is sent from an Inferior to the Superior, this shall be the inferior identifier as on the earlier ENROL message. This parameter shall be absent when CONFIRMED is sent from Decider to Terminator.

address-as-decider When the message is sent from a Decider to the Terminator, this shall be the address-as-decider of the Decider as on the BEGUN message (with the transaction identifier, this determines who the message is from). This parameter shall be absent when CONFIRMED is sent from an Inferior to Superior.

transaction identifier When the message is sent from a Decider to the Terminator, this shall be the transaction identifier as on the BEGUN message (i.e. the identifier of the Decider as a whole). This parameter shall be absent when CONFIRMED is sent from an Inferior to Superior

confirm received "true" if CONFIRMED is sent after receiving a CONFIRM message; "false" if an autonomous confirm decision has been made and either if no CONFIRM message has been received or the implementation cannot determine if CONFIRM has been received (due to loss of state information in a failure). This parameter shall be absent when CONFIRMED is sent from Decider to Terminator.

qualifiers standardised or other qualifiers.

Types of FAULT possible (sent to address-as-inferior)

General
InvalidSuperior – if Superior identifier is unknown
InvalidInferior – if no ENROL has been received for this address-as-inferior and identifier, or if RESIGN has been received from this Inferior.

1657	Note – A CONFIRMED message arriving before a CONFIRM message is
1658	sent, or after a CANCEL has been sent will occur when the Inferior has
1659	taken an autonomous decision and is not regarded as occurring in the wrong
1660	state. (The latter will cause a CONTRADICTION message to be sent.)

1661
1662   The form CONFIRMED/auto refers to a CONFIRMED message with "confirm
1663   received" = "false"; CONFIRMED/response refers to a CONFIRMED message
1664   with "confirm received" = "true". The unqualified form CONFIRMED refers to
1665   the message without an confirm received parameter, as used between Decider
1666   and Terminator.
1667
1668
1669 **CANCEL**
1670
1671 Sent by the Superior to an Inferior at any time before (and unless) CONFIRM has been sent.
1672
1673 Sent by a Terminator to a Decider at any time before REQUEST_CONFIRM has been sent.
1674

Parameter	Type
target address	BTP address
inferior identifier	Identifier
reply address	BTP address
transaction identifier	Identifier
inferiors-list	List of inferior handles
qualifiers	List of qualifiers

1675
1676   **target address** the address to which the CANCEL message is sent. When sent
1677   from Superior to Inferior, this will be the address-as-inferior from the ENROL
1678   message,. When sent from Terminator to Composer, this will be the decider-
1679   address from the BEGUN message .
1680
1681   **inferior identifier** When sent from Superior to Inferior, the inferior identifier as
1682   on the earlier ENROL message. This parameter shall be absent when sent from
1683   Terminator to Decider.
1684
1685   **reply address** When sent from Terminator to Decider, the address of the
1686   Terminator sending the CANCEL message. This parameter shall be absent when
1687   sent from Superior to Inferior.
1688
1689   **transaction identifier** When sent from Terminator to Decider, identifies the
1690   Decider and will be the transaction-identifier from the BEGUN message.. This
1691   parameter shall be absent when sent from Superior to Inferior.
1692

1693	**inferiors-list** When sent from Terminator to Composer, defines which of the
1694	Inferiors of this Composer are to be cancelled. This parameter shall be absent
1695	when sent from a Superior to an Inferior and when sent from a Terminator to a
1696	Coordinator.
1697	
1698	**qualifiers** standardised or other qualifiers.

1699

When sent to an Inferior, the effects of any operations associated with the Inferior should be
undone. If the Inferior had sent PREPARED, the Inferior is released from its promise to be
able to confirm the operations.

When sent to a Decider with the inferiors-list parameter is absent, the business transaction is
cancelled – this is propagated to any remaining Inferiors by issuing CANCEL to them. No
more Inferiors will be permitted to enrol.

When sent to a Composer, with the inferiors-list parameter present, only the Inferiors
identified in the inferiors-list are to be cancelled. Any other inferiors are unaffected by a
CANCEL/inferiors. Further Inferiors may be enrolled.

Note – A CANCEL/inferiors issued to a Cohesion Composer identifying all
of its currently enrolled Inferiors will leave the Cohesion 'empty', but
permitted to continue with new Inferiors, if any enrol.

Types of FAULT possible (sent to Superior address)

> *General*
> *UnknownTransaction* – if the transaction-identifier is unknown
> *InvalidInferior* – if inferior identifier is unknown, or an inferior-handle
> on the inferiors-list is unknown
> *WrongState* – if a CONFIRM has been received by this Inferior; if a
> REQUEST_CONFIRM has been received by this Composer.

The form CANCEL/whole refers to a CANCEL message sent to a Decider where the
"inferiors-list" parameter is absent. The form CANCEL/inferiors refers to a CANCEL
message sent to a Composer where the "inferiors-list" parameter is present. The unqualified
form CANCEL is used to refer to a CANCEL message sent from a Superior to an Inferior.

CANCELLED

Sent when the Inferior has applied (or is applying) cancellation of the operations associated
with the Inferior. CANCELLED is sent from Inferior to Superior in the following cases:

1. before (and instead of) sending PREPARED, to indicate the Inferior is unable to
   apply the operations in full and is cancelling all of them;

1739   2. in reply to CANCEL, regardless of whether PREPARED has been sent;
1740
1741   3. after sending PREPARED and then making and applying an autonomous
1742   decision to cancel.
1743
1744   4. in reply to CONFIRM_ONE_PHASE if the Inferiro decides to cancel the
1745   associated operations
1746
1747   As is specified in the state tables, cases 1, 2 and 3 are not always distinct in some
1748   circumstances of recovery and resending of messages.
1749
1750   CANCELLED is also sent by Decider to a Terminator in reply to REQUEST_CONFIRM if
1751   all Inferiors cancel without reporting hazards.
1752

Parameter	
target address	BTP address
superior identifier	Identifier
address-as-inferior	Set of BTP address
inferior identifier	Identifier
address-as-decider	BTP address
transaction-identifier	identifier
qualifiers	List of qualifiers

1753
1754   **target address** the address to which the CANCELLED is sent. When sent by an
1755   Inferior to a Superior, this will be the Superior address as on the CONTEXT
1756   message. When sent from a Decider to a Terminator it will be the reply address
1757   from the REQUEST_CONFIRM message.
1758
1759   **superior identifier** When the message is sent from an Inferior to the Superior,
1760   this shall be the superior identifier as on the CONTEXT message. This parameter
1761   shall be absent when CANCELLED is sent from Decider to Terminator.
1762
1763   **address-as-inferior** When the message is sent from an Inferior to the Superior,
1764   this shall be the address-as-inferior as on the earlier ENROL message (with the
1765   inferior identifier, this determines who the message is from). This parameter shall
1766   be absent when CANCELLED is sent from Decider to Terminator.
1767
1768   **inferior identifier** When the message is sent from an Inferior to the Superior, this
1769   shall be the inferior identifier as on the earlier ENROL message. This parameter
1770   shall be absent when CANCELLED is sent from Decider to Terminator.
1771
1772   **address-as-decider** When the message is sent from a Decider to the
1773   Terminator, this shall be the address-as-decider of the Decider as on the BEGUN
1774   message (with the transaction identifier, this determines who the message is

1775 from). This parameter shall be absent when CANCELLED is sent from an
1776 Inferior to Superior.
1777
1778 **transaction identifier** When the message is sent from a Decider to the
1779 Terminator, this shall be the transaction identifier as on the BEGUN message (i.e.
1780 the identifier of the Decider as a whole). This parameter shall be absent when
1781 CANCELLED is sent from an Inferior to Superior
1782
1783 **qualifiers** standardised or other qualifiers.
1784
1785 Types of FAULT possible (sent to address-as-inferior)
1786
1787 *General*
1788 *InvalidSuperior* – if Superior identifier is unknown
1789 *InvalidInferior* – if no ENROL has been received for this address-as-
1790 inferior and identifier, or if RESIGN has been received from this Inferior
1791 *WrongState* – if CONFIRM has been sent
1792

1793 Note – A CANCELLED message arriving before a CANCEL message is
1794 sent, or after a CONFIRM has been sent will occur when the Inferior has
1795 taken an autonomous decision and is not regarded as occurring in the wrong
1796 state. (The latter will cause a CONTRADICTION message to be sent.)

1797
1798
1799 ## CONFIRM_ONE_PHASE
1800
1801 Sent from a Superior to an enrolled Inferior, when there is only one such enrolled Inferior. In
1802 this case the two-phase exchange is not performed between the Superior and Inferior and the
1803 outcome decision for the operations associated with the Inferior is determined by the Inferior.
1804

Parameter	Type
target address	BTP address
inferior identifier	Identifier
report-hazard	boolean
qualifiers	List of qualifiers

1805
1806 **target address** the address to which the CONFIRM_ONE_PHASE message is
1807 sent This will be the address-as-inferior on the ENROL message.
1808
1809 **inferior identifier** The inferior identifier as on the earlier ENROL message for
1810 this Inferior.
1811

1812	**report hazard** Defines whether the superior wishes to be informed if a mixed
1813	condition occurs for the operations associated with the Inferior. If "report hazard"
1814	is "true", the Inferior will reply with HAZARD if a mixed condition occurs, or if
1815	the Inferior cannot determine that a mixed condition has not occurred. If "report
1816	hazard" is false, the Inferior will report only its own decision, regardless of
1817	whether that decision was correctly and consistently applied. Default is false.
1818	
1819	**qualifiers** standardised or other qualifiers.
1820	
1821	CONFIRM_ONE_PHASE can be issued by a Superior to an Inferior from whom
1822	PREPARED has been received (subject to the requirement that there is only one enrolled
1823	Inferior).
1824	
1825	Types of FAULT possible (sent to Superior address)
1826	
1827	*General*
1828	*InvalidInferior* – if inferior identifier is unknown
1829	*WrongState* – if a PREPARE has already been received from this
1830	Inferior
1831	

HAZARD

1832

1833

1834 Sent when the Inferior has either discovered a "mixed" condition: that is unable to correctly
1835 and consistently cancel or confirm the operations in accord with the decision (either the
1836 received decision of the superior or its own autonomous decision), or when the Inferior is
1837 unable to determine that a "mixed" condition has not occurred.
1838

1839 HAZARD is also used to reply to a CONFIRM_ONE_PHASE if the Inferior determines there
1840 is a mixed condition within its associated operations or is unable to determine that there is not
1841 a mixed condition.
1842

Parameter	Type
target address	BTP address
superior identifier	Identifier
address-as-inferior	Set of BTP address
inferior identifier	Identifier
Qualifiers	List of qualifiers

1843

1844	**target address** the address to which the MIXED is sent. This will be the
1845	superior address from the ENROL message.
1846	
1847	**superior identifier** The superior identifier as used on the ENROL message
1848	
1849	**address-as-inferior** The address-as-inferior as on the earlier ENROL message
1850	(with the inferior identifier, this determines who the message is from)

1851

1852   **inferior identifier** The inferior identifier as on the earlier ENROL message

1853

1854   **qualifiers** standardised or other qualifiers.

1855

1856   Types of FAULT possible (sent to address-as-inferior)

1857

1858   *General*
1859   *InvalidSuperior* – if Superior identifier is unknown
1860   *InvalidInferior* – if no ENROL has been received for this address-as-
1861   inferior and identifier, or if RESIGN has been received from this Inferior

1862

1863

1864   The form HAZARD/mixed refers to a HAZARD message with "level" = "mixed", the form
1865   HAZARD/possible refers to a HAZARD message with "level" = "possible".

1866

1867 **CONTRADICTION**

1868

1869   Sent by the Superior to an Inferior that has taken an autonomous decision contrary to the
1870   decision for the atom. This is detected by the Superior when the 'wrong' one of
1871   CONFIRMED or CANCELLED is received. CONTRADICTION is also sent in response to a
1872   HAZARD message.

1873

Parameter	Type
target address	BTP address
inferior identifier	Identifier
Qualifiers	List of qualifiers

1874

1875   **target address** the address to which the CONTRADICTION message is sent.
1876   This will be the address-as-inferior from the ENROL message.

1877

1878   **inferior identifier** The inferior identifier as on the earlier ENROL message for
1879   this Inferior.

1880

1881   **qualifiers** standardised or other qualifiers.

1882

1883   Types of FAULT possible (sent to Superior address)

1884

1885   *General*
1886   *InvalidInferior* – if inferior identifier is unknown
1887   *WrongState* – if neither CONFIRMED or CANCELLED has been sent
1888   by this Inferior

1889

1890 **SUPERIOR_STATE**

1891

Sent by a Superior as a query to an Inferior when

1. in the active state

2. there is uncertainty what state the Inferior has reached (due to recovery from previous failure or other reason).

Also sent by the Superior to the Inferior in response to a received INFERIOR_STATE, in particular states.

Parameter	Type
target address	BTP address
inferior identifier	Identifier
Status	*see below*
reply requested	Boolean
Qualifiers	List of qualifiers

target address the address to which the SUPERIOR_STATE message is sent. This will be the address-as-inferior from the ENROL message.

inferior identifier The inferior identifier as on the earlier ENROL message for this Inferior.

status states the current state of the Superior, in terms of its relation to this Inferior only.

status value	meaning
active	The relationship with the Inferior is in the active state from the perspective of the Superior; ENROLLED has been sent, PREPARE has not been sent and PREPARED has not been received (as far as the Superior knows)
prepared-received	PREPARED has been received from the Inferior, but no outcome is yet available
inaccessible	The state information for the Superior, or for its relationship with this Inferior, if it exists, cannot be accessed at the moment. This should be a transient condition
unknown	The Inferior is not known – it does not exist from the perspective of the Superior. The Inferior can treat this as an instruction to cancel any associated operations

Reply requested true, if SUPERIOR_STATE is sent as a query at the Superior's initiative; false, if SUPERIOR_STATE is sent in reply to a received

1915   INFERIOR_STATE or other message. Can only be true if status is active or
1916   prepared-received.
1917

1918   **qualifiers** standardised or other qualifiers.
1919

1920   The Inferior, on receiving SUPERIOR_STATE with reply requested = true, should reply in a
1921   timely manner by (depending on its state) repeating the previous message it sent or by
1922   sending INFERIOR_STATE with the appropriate status value.
1923

1924   A status of unknown shall only be sent if it has been determined for certain that the Superior
1925   has no knowledge of the Inferior, or (equivalently) it can be determined that the relationship
1926   with the Inferior was cancelled. If there could be persistent information corresponding to the
1927   Superior, but it is not accessible from the entity receiving an INFERIOR_STATE/*/y (or
1928   other) message targeted to the Superior or that entity cannot determine whether any such
1929   persistent information exists or not, the response shall be Inaccessible.
1930

1931   SUPERIOR_STATE/unknown is also used as a response to messages, other than
1932   INFERIOR_STATE/*/y that are received when the Inferior is not known (and it is known
1933   there is no state information for it).
1934

1935   The form SUPERIOR_STATE/abcd refers to a SUPERIOR_STATE message status having a
1936   value equivalent to "abcd" (for active, prepared-received, unknown and inaccessible) and
1937   with "reply requested" = "false". SUPERIOR_STATE/abcd/y refers to a similar message, but
1938   with "reply requested" = "true". The form SUPERIOR_STATE/*/y refers to a
1939   SUPERIOR_STATE message with "reply requested" = "true" and any value for status.
1940

1941

1942   **INFERIOR_STATE**
1943

1944   Sent by a Inferior as a query when in the active state to a Superior, when (due recovery from
1945   previous failure or other reason) there is uncertainty what state the Superior has reached.
1946

1947   Also sent by the Inferior to the Superior in response to a received SUPERIOR_STATE, in
1948   particular states.
1949

Parameter	Type
target address	BTP address
superior identifier	Identifier
address-as-inferior	BTP address
inferior identifier	Identifier
Status	*see below*
reply requested	Boolean
Qualifiers	List of qualifiers

1950

1951	**target address** the address to which the INFERIOR_STATE is sent. This will
1952	be the target address as used the original ENROL message.
1953	
1954	**superior identifier** The superior identifier as used on the ENROL message
1955	
1956	**address-as-inferior** The address-as-inferior as on the ENROL message (with the
1957	inferior identifier, this determines who the message is from)
1958	
1959	**inferior identifier** The inferior identifier as on the ENROL message
1960	
1961	**status** states the current state of the Inferior for the atomic business transaction,
1962	which corresponds to the last message sent to the Superior by (or in the case of
1963	ENROL for) the Inferior
1964	

status value	meaning/previous message sent
active	The relationship with the Superior is in the active state from the perspective of the Inferior; ENROL has been sent, a decision to send PREPARED has not been made.
inaccessible	The state information for the relationship with the Superior, if it exists, cannot be accessed at the moment. This should be a transient condition
unknown	The Inferior is not known – it does not exist from the perspective of the Superior. The Inferior can be treated as cancelled

1965	
1966	**reply requested** "true" if INFERIOR_STATE is sent as a query at the
1967	Superior's initiative; "false" if INFERIOR_STATE is sent in reply to a received
1968	SUPERIOR_STATE or other message. Can only be "true" if "status" is "active"
1969	or "prepared-received". Can only be "true" if "status" is "active".
1970	
1971	**qualifiers** standardised or other qualifiers.
1972	
1973	The Superior, on receiving INFERIOR_STATE with "reply requested" = "true", should reply
1974	in a timely manner by (depending on its state) repeating the previous message it sent or by
1975	sending SUPERIOR_STATE with the appropriate status value.
1976	
1977	A status of "unknown" shall only be sent if it has been determined for certain that the Inferior
1978	has no knowledge of a relationship with the Superior. If there could be persistent information
1979	corresponding to the Superior, but it is not accessible from the entity receiving an
1980	SUPERIOR_STATE/*/y (or other) message targtted on the Inferior or the entity cannot
1981	determine whether any such persistent information exists, the response shall be
1982	"inaccessible".
1983	
1984	INFERIOR_STATE/unknown is also used as a response to messages, other than
1985	SUPERIOR_STATE/*/y that are received when the Inferior is not known (and it is known
1986	there is no state information for it).
1987	

1988   A SUPERIOR_STATE/INFERIOR_STATE exchange that determines that one or both sides
1989   are in the active state does not require that the Inferior be cancelled (unlike some other two-
1990   phase commit protocols). The relationship between Superior and Inferior, and related
1991   application elements may be continued, with new application messages carrying the same
1992   CONTEXT. Similarly, if the Inferior is prepared but the Superior is active, there is no
1993   required impact on the progression of the relationship between them.
1994
1995   The form INFERIOR_STATE/abcd refers to a INFERIOR_STATE message status having a
1996   value equivalent to "abcd" (for active, unknown and inaccessible) and with "reply requested"
1997   = "false". INFERIOR_STATE/abcd/y refers to a similar message, but with "reply requested"
1998   = "true". The form INFERIOR_STATE/*/y refers to a INFERIOR_STATE message with
1999   "reply requested" = "true" and any value for status.
2000
2001   **REQUEST_CONFIRM**
2002
2003   Sent from a Terminator to a Decider to request confirmation of the business transaction. If the
2004   business transaction is a Cohesion, the confirm-set is specified by the "inferiors-list"
2005   parameter.
2006

Parameter	Type
target address	BTP address
reply address	BTP address
transaction identifier	Identifier
inferiors-list	List of inferior handles
Report hazard	boolean
Qualifiers	List of qualifiers

2007
2008   **target address**   the address to which the REQUEST_CONFIRM message is
2009   sent. This will be the address-as-decider on the BEGUN message.
2010
2011   **reply address**   the address of the Terminator sending the REQUEST_CONFIRM
2012   message.
2013
2014   **transaction identifier**   identifies the Decider. This will be the transaction-
2015   identifier from the BEGUN message.
2016
2017   **inferiors-list**   defines which Inferiors enrolled with the Decider, if it is a
2018   Cohesion Composer, are to be confirmed. Shall be absent if the Decider is an
2019   Atom Coordinator.
2020
2021   **report hazard**   Defines whether the Terminator wishes to be informed of hazard
2022   events and contradictory decisions within the business transaction. If "report
2023   hazard" is "true", the receiver will wait until responses (CONFIRMED,
2024   CANCELLED or HAZARD) have been received from all of its inferiors,
2025   ensuring that any hazard events are reported. If "report hazard" is "false", the

2026   Decider will reply with CONFIRMED or CANCELLED as soon as the decision
2027   for the transaction is known.

2028

2029   **qualifiers** standardised or other qualifiers.

2030

2031 If the "inferiors-list" parameter is present, the Inferiors identified shall be the "confirm-set" of
2032 the Cohesion. It the parameter is absent and the business transaction is a Cohesion, the
2033 "confirm-set" shall be all remaining Inferiors. If the business transaction is an Atom, the
2034 "confirm-set" is automatically all the Inferiors.

2035

2036 Any Inferiors from which RESIGN is received are not counted in the confirm-set.

2037

2038 If PREPARED has not been received from any Inferiors in the confirm-set, PREPARE shall
2039 be issued to them.

2040

2041 A confirm decision may be made only if PREPARED has been received from all Inferiors in
2042 the "confirm-set". The making of the decision shall be persistent (and if it is not possible to
2043 persist the decision, it is not made). If there is only one remaining Inferior in the "confirm
2044 set", CONFIRM_ONE_PHASE may be sent to it.

2045

2046 All remaining Inferiors that are not in the confirm set shall be cancelled.

2047

2048 If a confirm decision is made and "report-hazard" was "false", a CONFIRMED message shall
2049 be sent to the "reply-address".

2050

2051 If a cancel decision is made and "report-hazard" was "false", a CANCELLED message shall
2052 be sent to the "reply-address".

2053

2054 If "report-hazard" was "true" and any HAZARD or contradictory message was received (i.e.
2055 CANCELLED from an Inferior in the confirm-set or CONFIRMED from an Inferior not in
2056 the confirm-set), an INFERIOR_STATUSES reporting the status for all Inferiors shall be sent
2057 to the "reply-address".

2058

2059 Types of FAULT possible (sent to reply address)

2060

2061   *General*
2062   *UnknownTransaction* – if the transaction-identifier is unknown
2063   *InvalidInferior* – if an inferior handle in the inferiors-list is unknown
2064   *WrongState* – if a CANCEL/whole has already been received .

2065

2066 The form REQUEST_CONFIRM/whole refers to a REQUEST_ CONFIRM message where
2067 the "inferiors-list" parameter is absent. The form REQUEST_ CONFIRM /inferiors refers to
2068 a REQUEST_ CONFIRM message where the "inferiors-list" parameter is present.

2069

2070 **REQUEST_STATUSES**

2071

2072 Sent to a Decider to ask it to report the status of its Inferiors with an
2073 INFERIOR_STATUSES message.

2074

Parameter	Type
target address	BTP address
reply address	BTP address
transaction identifier	Identifier
inferiors-list	List of inferior handles
Qualifiers	List of qualifiers

2075

2076   **target address** the address to which the REQUEST_ STATUS message is
2077   sent..This will be the address-as-decider from the BEGUN message.
2078
2079   **reply address** the address to which the replying INFERIOR_STATUSES is to
2080   be sent
2081
2082   **transaction identifier** identifies the Decider. This will be the transaction-
2083   identifier from the BEGUN message.
2084
2085   **inferiors-list** defines which inferiors enrolled with the Composer or Coordinator
2086   are to be included in the INFERIOR_STATUSES. If the list is absent, the status
2087   of all enrolled inferiors will be reported.
2088
2089   **qualifiers** standardised or other qualifiers.
2090
2091   Types of FAULT possible (sent to reply-address)
2092
2093   *General*
2094
2095   The form REQUEST_STATUSES/whole refers to a REQUEST_STATUS with the inferiors-
2096   list absent. The form REQUEST_STATUS/inferiors refers to a REQUEST_STATUS with
2097   the inferiors-list present.
2098

INFERIOR_STATUSES

2099
2100
2101   Sent by a Decider to report the status of all or some of its inferiors in response to a
2102   REQUEST_STATUSES, PREPARE, CANCEL/inferiors and REQUEST_CONFIRM with
2103   "report-hazard" = "true".
2104

Parameter	Type
target address	BTP address
address-as-decider	BTP address
transaction-identifier	identifier
status-list	Set of Status items - see below

Parameter	Type
general-qualifiers	List of qualifiers

target address the address to which the INFERIOR_STATUSES is sent. This will be the reply address on the received message

address-as-decider The address-as-decider of the Decider as on the BEGUN message (with the transaction identifier, this determines who the message is from)

transaction identifier The transaction identifier as on the BEGUN message (i.e. the identifier of the Decider as a whole)

status-list contains a number of Status-items, each reporting the status of one of the inferiors of the Decider. The fields of a Status-item are

Field	Type
Inferior-handle	Inferior handle, identifying which inferior this Status-item contains information for.
status	One of the status values below (these are a subset of those for STATUS)
qualifiers	A list of qualifiers as received from the particular inferior or associated with the inferior in earlier messages (e.g. an Inferior name qualifier).

The status value reports the current status of the particular inferior, as known to the Composer or Coordinator. Values are:

status value	Meaning
active	The Inferior is enrolled
resigned	RESIGNED has been received from the Inferior
preparing	PREPARE has been sent to the inferior, none of PREPARED, RESIGNED, CANCELLED, HAZARD have been received
prepared	PREPARED has been received
autonomously confirmed	CONFIRMED/auto has been received, no completion message has been sent
autonomously cancelled	PREPARED had been received, and since then CANCELLED has been received but no completion message has been sent
confirming	CONFIRM has been sent, no outcome reply has been received

status value	Meaning
confirmed	CONFIRMED/response has been received
cancelling	CANCEL has been sent, no outcome reply has been received
cancelled	CANCELLED has been received, and PREPARED was not received previously
cancel-contradiction	Confirm had been ordered (and may have been sent), but CANCELLED was received
confirm-contradiction	Cancel had been ordered (and may have been sent) but CONFIRM/auto was received
hazard	A HAZARD message has been received

General qualifiers standardised or other qualifiers applying to the INFERIOR_STATUSES as a whole. Each Status-item contains a "qualifiers" field containing qualifiers applying to (and received from) the particular Inferior.

If the inferiors-list parameter was present on the received message, only the inferiors identified by that parameter shall have their status reported in status-list of this message. If the inferiors-list parameter was absent, the status of all enrolled inferiors shall be reported, except that an inferior that had been reported as *cancelled* or *resigned* on a previous INFERIOR_STATUSES message **may** be omitted (sender's option).

REQUEST_STATUS

Sent to an Inferior or to a Decider to ask it to reply with STATUS.

Parameter	Type
target address	BTP address
reply address	BTP address
inferior identifier	Identifier
transaction-identifier	Identifier
Qualifiers	List of qualifiers

target address the address to which the REQUEST_ STATUS message is sent..If the target is an Inferior, this will be the address-as-inferior on the ENROL message. If the target is a Decider, this will be the address-as-decider on the BEGUN message.

reply address the address to which the replying STATUS should be sent

inferior identifier If the target is an Inferior, the "inferior-identifier" on the ENROL message. If the target is a Decider, this parameter shall be absent.

2148

2149   **transaction-identifier** If the target is a Decider, the "transaction-identifier" on
2150   the BEGUN message. If the target is an Inferior, this parameter shall be absent.

2151

2152   **qualifiers** standardised or other qualifiers.

2153

2154 Types of FAULT possible (sent to reply address)

2155

2156   *General*

2157

2158

2159 **STATUS**

2160

2161 Sent by a Inferior or Decider in reply to a REQUEST_STATUS, reporting the overall state of
2162 the transaction tree node represented by the Inferior or Decider.

2163

Parameter	Type
target address	BTP address
address-as-inferior	BTP address
inferior identifier	Identifier
address-as-decider	BTP address
transaction-identifier	Identifier
status	See below
qualifiers	List of qualifiers

2164

2165   **target address** the address to which the STATUS is sent. This will be the reply
2166   address on the REQUEST_STATUS message

2167

2168   **address-as-inferior** If the sender is an Inferior, the address-as-inferior as on the
2169   ENROL message (with the inferior-identifier, this determines who the message is
2170   from). If the sender is a Decider, this parameter shall be absent

2171

2172   **inferior-identifier** If the sender is an Inferior, the inferior-identifier as on the
2173   ENROL message. If the sender is a Decider, this parameter shall be absent.

2174

2175   **address-as-decider** If the sender is a Decider, the address-as-decider on the
2176   BEGUN message (with the "transaction-identifier", this determines who the
2177   message is from). If the sender is an Inferior, this parameter shall be absent.

2178

2179   **transaction-identifier** If the sender is a Decider, the transaction identifier as on
2180   the BEGUN message. If the sender is an Inferior, this parameter shall be absent.

2181

2182   **status** states the current status of the transaction tree node represented by the
2183   sender.

2184

status value	Meaning from Inferior	Meaning from Decider
Created	The Inferior exists (and is addressable) but it has not been enrolled with a Superior	Not applicable
Enrolling	ENROL has been sent, but ENROLLED is awaited	Not applicable
Active	The Inferior is enrolled	New enrolment of inferiors is possible; no decision has been made.
Resigning	RESIGN has been sent; RESIGNED is awaited	Not applicable
Resigned	RESIGNED has been received	Not applicable
Preparing	PREPARE has been received; PREPARED has not been sent	Not applicable
Prepared	PREPARED has been sent; no outcome has been received or autonomous decision made	Not applicable
Confirming	CONFIRM has been received; CONFIRMED/response has not bee sent	Confirm decision has been made but responses from inferiors are pending
Confirmed	CONFIRMED/response has been sent	CONFIRMED has been sent
Cancelling	CANCEL has been received or auto-cancel has been decided	Cancel decision has been made but responses from inferiors are pending
Cancelled	CANCELLED has been sent	CANCELLED has been sent
cancel-contradiction	Autonomous cancel decision was made, CONFIRM received; CONTRADICTION has not been received	Not applicable
confirm-contradiction	Autonomous confirm decision was made, CANCEL received; CONTRADICTION has not been received	Not applicable
Hazard	A hazard has been discovered; CONTRADICTION has not been received	A hazard has been reported from at least one Inferior
Contradicted	CONTRADICTION has been received	Not applicable

status value	Meaning from Inferior	Meaning from Decider
Unknown	No state information for the identifier exists; no such Inferior exists	No state information for the transaction identifier exists; no such Decider exists
Inaccessible	There may be state information for this identifier but it cannot be reached/existence cannot be determined	There may be state information for this identifier but it cannot be reached/existence cannot be determined

2185
2186   **qualifiers** standardised or other qualifiers.
2187

2188 ## REDIRECT

2189

2190 Sent when the address previously given for a Superior or Inferior is no longer valid and the
2191 relevant state information is now accessible with a different address (but the same superior or
2192 inferior identifier).
2193

Parameter	Type
target address	BTP address
superior identifier	Identifier
inferior identifier	Identifier
old address	Set of BTP addresses
new address	Set of BTP addresses
qualifiers	List of qualifiers

2194
2195   **target address** the address to which the REDIRECT is sent. This may be the
2196 reply address from a received message or the address of the opposite side
2197 (superior/inferior) as given in a CONTEXT or ENROL message
2198
2199   **superior identifier** The superior identifier as on the CONTEXT message and
2200 used on an ENROL message. (present only if the REDIRECT is sent from the
2201 Inferior).
2202
2203   **inferior identifier** The inferior identifier as on the ENROL message
2204
2205   **old address** The previous address of the sender of REDIRECT. A match is
2206 considered to apply if any of the old addresses match one that is already known.
2207
2208   **new address** The (set of alternatives) new addresses to be used for messages
2209 sent to this entity.
2210
2211   **qualifiers** standardised or other qualifiers.
2212

2213	If the actor whose address is changed is a Inferior, the new address value replaces
2214	the address-as-inferior as present in the ENROL.
2215	
2216	If the actor whose address is changed is a Superior, the new address value
2217	replaces the Superior address as present in the CONTEXT message (or as present
2218	in any other mechanism used to establish the Superior:Inferior relationship).
2219	
2220	

2221 **FAULT**

2222

2223   Sent in reply to various messages to report an error condition

2224

Parameter	Type
target address	BTP address
superior identifier	Identifier
inferior identifier	Identifier
fault type	See below
fault data	See below
qualifiers	List of qualifiers

2225

2226   **target address** the address to which the FAULT is sent. This may be the reply
2227   address from a received message or the address of the opposite side
2228   (superior/inferior) as given in a CONTEXT or ENROL message

2229

2230   **superior identifier** the superior identifier as on the CONTEXT message and as
2231   used on the ENROL message (present only if the FAULT is sent to the superior).

2232

2233   **inferior identifier** the inferior identifier as on the ENROL message (present only
2234   if the FAULT is sent to the inferior)

2235

2236   **fault type** identifies the nature of the error, as specified for each of the main
2237   messages.

2238

2239   **fault data** information relevant to the particular error. Each fault type defines the
2240   content of the fault data:

2241

fault type	meaning	fault data
General	Any otherwise unspecified problem	Free text explanation
UnknownParameter	A BTP message has been received with an unrecognised parameter	Free text explanation
WrongState	The message has arrived when the recipient is in an invalid	

	the recipient is in an invalid state.	
CommunicationFailure	Any fault arising from the carrier mechanism and communication infrastructure.	Determined by the carrier mechanism and binding specification
InvalidSuperior	The received identifier is not known or does not identify a known Superior	The identiifier
DuplicateInferior	An inferior with the same address and identifier is already enrolled with this Superior	The identiifier
InvalidInferior	The Superior is known but the Inferior identified by the address-as-inferior and identifier are not enrolled in it	The Inferior Identity (address-as-inferior and identifier)
UnsupportedQualifier	A qualifier has been received that is not recognised and on which "must-be-Understood" is "true".	Qualifier group and name

qualifiers standardised or other qualifiers.

Note – If the carrier mechanism used for the transmission of BTP messages is capable delivering messages in a different order than they were sent in, the "WrongState" FAULT is not sent and should be ignored if received.

Standard qualifiers

The following qualifiers are expected to be of general use to many applications and environments. The URI "`urn:oasis:names:tc:BTP:qualifiers`" is used in the Qualifier group value for the qualifiers defined here.

Transaction timelimit

The transaction timelimit allows the Superior (or an application element initiating the business transaction) to indicate the expected length of the active phase, and thus give an indication to the Inferior of when it would be appropriate to initiate cancellation if the active phase appears to continue too long. The time limit ends (the clock stops) when the Inferior decides to be prepared and issues PREPARED to the Superior.

It should be noted that the expiry of the time limit does not change the permissible actions of the Inferior. At any time prior to deciding to be prepared (for an Inferior), the Inferior is

2266	**permitted** to initiate cancellation for internal reasons. The timelimit gives an indication to the	
2267	entity of when it will be useful to exercise this right.	

2269 The qualifier is propagated on a CONTEXT message.

2271 The "Qualifier name" shall be "`transaction-timelimit`".

2273 The "Content" shall contain the following field:

Content field	Type
Timelimit	Integer

2276 **Timelimit** indicates the maximum (further) duration, expressed as whole seconds from the
2277 time of transmission of the containing CONTEXT, of the active phase of the business
2278 transaction.

Inferior timeout

2282 This qualifier allows an Inferior to limit the duration of its "promise", when sending
2283 PREPARED, that it will maintain the ability to confirm or cancel the effects of all associated
2284 operations. Without this qualifier, an Inferior is expected to retain the ability to confirm or
2285 cancel indefinitely. If the timeout does expire, the Inferior is released from its promise and
2286 can apply the decision indicated in the qualifier.

2288 It should be noted that BTP recognises the possibility that an Inferior may be forced to apply
2289 a confirm or cancel decision before the CONFIRM or CANCEL is received and before this
2290 timeout expires (or if this qualifier is not used). Such a decision is termed a heuristic decision,
2291 and (as with other transaction mechanisms), is considered to be an exceptional event. As with
2292 heuristic decisions, the taking of an autonomous decision by a Inferior **subsequent** to the
2293 expiry of this timeout, is liable to cause contradictory decisions across the business
2294 transaction. BTP ensures that at least the occurrence of such a contradiction will be
2295 (eventually) reported to the Superior of the business transaction. BTP treats "true" heuristic
2296 decisions and autonomous decisions after timeout the same way – in fact, the expiry in this
2297 timeout does not cause a qualitative (state table) change in what can happen, but rather a step
2298 change in the probability that it will.

2300 The expiry of the timeout does not strictly require that the Inferior immediately invokes the
2301 intended decision, only that is at liberty to do so. An implementation may choose to only
2302 apply the decision if there is contention for the underlying resource, for example.
2303 Nevertheless, Superiors are recommended to avoid relying on this and ensure decisions for
2304 the business transaction are made before these timeouts expire (and allow a margin of error
2305 for network latency etc.).

2307 The qualifier may be present on a PREPARED message. If the PREPARED message has the
2308 "default is cancel" parameter "true", then the "IntendedDecision" field of this qualifier shall
2309 have the value "cancel".

2311   The "Qualifier name" shall be "`inferior-timeout`".
2312
2313   The "Content" shall contain the following fields:
2314

Content field	Type
Timeout	Integer
IntendedDecision	"confirm" or "cancel"

2315
2316   **Timeout** indicates how long, expressed as whole seconds from the time of transmission of the
2317   carrying message, the Inferior intends to maintain its ability to either confirm or cancel the
2318   effects of the associated operations, as ordered by the receiving Superior.
2319
2320   **IntendedDecision** indicates which outcome will be applied, if the timeout completes and an
2321   autonomous decision is made.
2322
2323 **Minimum inferior timeout**
2324
2325   This qualifier allows a Superior to constrain the Inferior timeout qualifier received from the
2326   Inferior. If a Superior knows that the decision for the business transaction will not be
2327   determined for some period, it can require that Inferiors do not send PREPARED messages
2328   with Inferior timeouts that would expire before then. An Inferior that is unable or unwilling to
2329   send a PREPARED message with a longer (or no) timeout **should** cancel, and reply with
2330   CANCELLED.
2331
2332   The qualifier may be present on a CONTEXT, ENROLLED or PREPARE message. If
2333   present on more than one, and with different values of the MinimumTimeout field, the value
2334   on ENROLLED shall prevail over that on CONTEXT and the value on PREPARE shall
2335   prevail over either of the others.
2336
2337   The "Qualifier name" shall be "`minimum-inferior-timeout`".
2338
2339   The "Content" shall contain the following field:
2340

Content field	Type
MinimumTimeout	Integer

2341
2342   **Minimum Timeout** is the minimum value of timeout, expressed as whole seconds, that will be
2343   acceptable in the Inferior timout qualifier on an answering PREPARED message.
2344
2345 **Inferior name**
2346
2347   This qualifier allows an Enroller to supply a name for the Inferior that will be visible on
2348   INFERIOR_STATUSES and thus allow the Terminator to determine which Inferior (of the
2349   Composer or Coordinator) is related to which application work. This is in addition to the
2350   "inferior handle" field. The name can be human-readable and can also be used in fault
2351   tracing, debugging and auditing.

2352

2353 The name is never used by the BTP actors themselves to identify each other or to direct
2354 messages. (The BTP actors use the addresses and the identifiers in the message parameters
2355 for those purposes.)

2356

2357 This specification makes no requirement that the names are unambiguous within any scope
2358 (unlike the "inferior-handle" on ENROLLED and BEGUN, which is required to be
2359 unambiguous within the scope of the Decider). Other specifications, including those defining
2360 use of BTP with a particular application may place requirements on the use and form of the
2361 names. (This may include reference to information passed in application messages or in other,
2362 non-standardised, qualifiers.)

2363

2364 The qualifier may be present on BEGIN, ENROL and in the "qualifiers" field of a Status-item
2365 in INFERIOR_STATUSES. It is present on BEGIN only if there is a related CONTEXT; if
2366 present, the same qualifier value **should** be included in the consequent ENROL. If
2367 INFERIOR_STATUSES includes a Status-item for an Inferior whose ENROL had an
2368 inferior-name qualifier, the same qualifier value **should** be included in the Status-item.

2369

2370 The "Qualifier -name" shall be "`inferior-name`"

2371

2372 The "Content" shall contain the following fields:

2373

Content field	Type
inferior-name	String

2374
2375 **Inferior name** the name assigned to the enrolling Inferior.
2376

State Tables

Explanation of the state tables

The state tables deal with the state transitions of the Superior and Inferior roles and which message can be sent and received in each state. The state tables directly cover only a single, bi-lateral Superior:Inferior relationship. The interactions between, for example, multiple Inferiors of a single Superior that will apply the same decision to all or some (of them , are dealt with in the definitions of the "decision" events which also specify when changes are made to persistent state information (see below).

There are two state tables, one for Superior, one for Inferior. States are identified by a letter-digit pair, with upper-case letters for the superior, lower-case for the inferior. The same letter is used to group states which have the same, or similar, persistent state, with the digit indicating volatile state changes or minor variations. Corresponding upper and lower-case letters are used to identify (approximately) corresponding Superior and Inferior states.

The Inferior table includes events occurring both at the Inferior as such and at the associated Enroller, as the Enroller's actions are constrained by and constrain the Inferior role itself.

Status queries

In BTP the messages SUPERIOR_STATE and INFERIOR_STATE are available to prompt the peer to report its current state by repeating the previous message (when this is allowed) or by sending the other *_STATE message. The "reply_requested" parameter of these messages distinguishes between their use as a prompt and as a reply. An implementation receiving a *_STATE message with "reply_requested" as "true" is not required to reply immediately – it may choose delay any reply until a decision event occurs and then send the appropriate new message (e.g. on receiving INFERIOR_STATE/prepared/y while in state E1, a superior is permitted to delay until it has performed "decide to confirm" (or "decide to cancel"). However, this may cause the other side to repeatedly send interrogatory *_STATE messages.

Note that a Superior (or some entity standing in for a now-extinct Superior) uses SUPERIOR_STATE/unknown to reply to messages received from an Inferior where the Superior:Inferior relationship is in an unknown (using state "Y1"). The *_STATE messages with a "state" value "inaccessible" can be used as a reply when **any** message is received and the implementation is temporarily unable to determine whether the relationship is known or what the state is. Other than these cases, the *_STATE messages with "reply requested" equal to "false" are only sent when the other message with "reply requested" equal to "true" has been received and no other message has been sent.

Decision events

The persistent state changes (equivalent to logging in a regular transaction system) and some other events are modelled as "decision events" (e.g. "decide to confirm", "decide to be prepared"). The exact nature of the real events and changes in an implementation that are modelled by these events depends on the position of the Superior or Inferior within the business transaction and on features of the implementation (e.g. making of a persistent record

2424 of the decision means that the information will survive at least some failures that otherwise
2425 lose state information, but the level of survival depends on the purpose of the
2426 implementation). Table 2 and Table 3 define the decision events.
2427
2428 In some cases, an implementation may not need to make an active change to have a persistent
2429 record of a decision, provided that the implementation will restore itself to the appropriate
2430 state on recovery. For example, an (inferior) implementation that "decided to be prepared",
2431 and recorded a timeout (to cancel) in the persistent information for that decision (signalled via
2432 the appropriate qualifier on PREPARED), could treat the presence of an expired record as a
2433 record of "decide to cancel autonomously", provided it always updated such a record as part
2434 of the "apply ordered confirmation" decision event.
2435
2436 The Superior event "decide to prepare" is considered semi-persistent. Since the sending of
2437 PREPARE indicates that the application exchange (to associate operations with the Inferior)
2438 is complete, it is not meaningful for the Superior:Inferior relationship to revert to an earlier
2439 state corresponding to an incomplete application exchange. However, implementations are
2440 not required to make the sending of PREPARE persistent in terms of recovery – a Superior
2441 that experiences failure after sending PREPARE may, on recovery, have no information
2442 about the transaction, in which case it is considered to be in the completed state (Z), which
2443 will imply the cancellation of the Inferior and its associated operations.
2444
2445 Where a Superior is itself an Inferior (to another Superior entity), in a hierarchic tree, its
2446 "decide to confirm" and "decide to cancel" decisions will in fact be the receipt of a
2447 CONFIRM or CANCEL instruction from its own Superior, without necessary change of local
2448 persistent information (which would combine both superior and inferior information, pointing
2449 both up and down the tree).
2450
2451
2452 ## Disruptions – failure events
2453
2454 Failure events are modelled as "disruption". A failure and the subsequent recovery will (or
2455 may) cause a change of state. The disruption events in the state tables model different extents
2456 of loss of state information. An implementation is not required to exhibit all the possible
2457 disruption events, but it is not allowed to exhibit state transitions that do not correspond to a
2458 possible disruption.
2459
2460 In addition to the disruption events in the tables, there is an implicit "disruption 0" event,
2461 which involves possible interruption of service and loss of messages in transit, but no change
2462 of state (either because no state information was lost, or because recovery from persistent
2463 information restores the implementation to the same state). The "disruption 0" event would
2464 typically be an appropriate abstraction for a communication failure.
2465
2466 ## Invalid cells and assumptions of the communication mechanism
2467
2468 The empty cells in state table represent events that cannot happen. For events corresponding
2469 to sending a message or any of the decision events, this prohibition is absolute – e.g. a
2470 conformant implementation in the Superior active state "B1" will not send CONFIRM. For

| 2471 | events corresponding to receiving a message, the interpretation depends on the properties of |
| 2472 | the underlying communications mechanism. |

2474 For all communication mechanisms, it is assumed that
- a) the two directions of the Superior:Inferior communication are not synchronised – that is messages travelling in opposite directions can cross each other to any degree; any number of messages may be in transit in either direction; and
- b) messages may be lost arbitrarily

If the communication mechanisms guarantee ordered delivery (i.e. that messages, if delivered at all, are delivered to the receiver in the order they were sent) , then receipt of a message in a state where the corresponding cell is empty indicates that the far-side has sent a message out of order – a FAULT message with the Fault Type "WrongState" can be returned.

If the communication mechanisms cannot guarantee ordered delivery, then messages received where the corresponding cell is empty should be ignored. Assuming the far-side is conformant, these messages can assumed to be "stale" and have been overtaken by messages sent later but already delivered. (If the far-side is non-conformant, there is a problem anyway).

Meaning of state table events

The tables in this section define the events (rows) in the state tables. Table 1 defines the events corresponding to sending or receiving BTP messages and the disruption events. Table 2 describes the decision events for an Inferior, Table 3 those for a Superior.

The decision events for a Superior, defined in Table 3 cannot be specified without reference to other Inferiors to which it is Superior and to its relation with the application or other entity that (acting ultimately on behalf of the application) drives it.

The term "remaining Inferiors" are any actors to which this endpoint is Superior and which are to be treated as an atomic decision unit with (and thus including) the Inferior on this relationship. If the CONTEXT for this Superior:Inferior relationship had a "superior type" of "atom", this will be all Inferiors established with same Superior address and Superior identifier except those from which RESIGN has been received. If the CONTEXT had "superior type" of "cohesion", the "remaining Inferiors" excludes any that it has been determined will be cancelled, as well as any that have resigned – in other words it includes only those for which a confirm decision is still possible or has been made. The determination of exactly which Inferiors are "remaining Inferiors" in a cohesion is determined, in some way, by the application. The term "Other remaining Inferiors" excludes this Inferior on this relationship. A Superior with a single Inferior will have no "other remaining Inferiors".

In order to ensure that the confirmation decision **is** delivered to all remaining Inferiors, despite failures, the Superior must persistently record which these Inferiors are (i.e. their addresses and identifiers). It must also either record that the decision is confirm, or ensure that the confirm decision (if there is one) is persistently recorded somewhere else, and that it will be told about it. This latter would apply if the Superior were also BTP Inferior to another entity which persisted a confirm decision (or recursively deferred it still higher). However,

2519	since there is no requirement that the Superior be also a BTP Inferior to any other entity, the
2520	behaviour of asking another entity to make (and persist) the confirm decision is termed
2521	"offering confirmation" - the Superior offers the possible confirmation of itself, and its
2522	remaining Inferiors to some other entity. If that entity (or something higher up) then does
2523	make and persist a confirm decision, the Superior is "instructed to confirm" (which is
2524	equivalent BTP CONFIRM).

2525

2526	The application, or an entity acting indirectly on behalf of the application, may request a
2527	Superior to prepare an Inferior (or all Inferiors). This typically implies that there will be no
2528	more operations associated with the Inferior. Following a request to prepare all remaining
2529	Inferiors, the Superior may offer confirmation to the entity that requested the prepare. (If the
2530	Superior is also a BTP Inferior, its superior can be considered an entity acting on behalf of the
2531	application.)

2532

2533	The application, or an entity acting indirectly on behalf of the application, may also request
2534	confirmation. This means the Superior is to attempt to make and persist a confirm decision
2535	itself, rather than offer confirmation.

2536

2537

2538 **Table 1 : send, receive and disruption events**

Event name	Meaning
send/receive ENROL/rsp-req	send/receive ENROL with reply-requested = true
send/receive ENROL/no-rsp-req	send/receive ENROL with reply-requested = false
send/receive RESIGN/rsp-req	send/receive RESIGN with reply-requested = true
send/receive RESIGN/no-rsp-req	send/receive RESIGN with reply-requested = false
send/receive PREPARED	send/receive PREPARED, with default-cancel = false
send/receive PREPARED/cancel	send/receive PREPARED, with default-cancel = true
send/receive CONFIRMED/auto	send/receive CONFIRMED, with confirm-received = true
send/receive CONFIRMED/response	send/receive CONFIRMED, with confirm-received = false
send/receive HAZARD	send/receive HAZARD
send/receive INF_STATE/***/y	send/receive INFERIOR_STATE with status *** and reply-requested = true
send/receive INF_STATE/***	send/receive INFERIOR_STATE with status *** and reply-requested = false
send/receive SUP_STATE/***/y	send/receive SUPERIOR_STATE with status *** and reply-requested = true ("prepared-rcvd" represents "prepared-received")

Event name	Meaning
send/receive SUP_STATE/***	send/receive SUPERIOR_STATE with status *** and reply-requested = false ("prepared-rcvd" represents "prepared-received")
disruption ***	Loss of state – new state is state applying after any local recovery processes complete

2539

2540 **Table 2 : Decision events for Inferior**

Event name	Meaning
decide to resign	• Any associated operations have had no effect (data state is unchanged)).
decide to be prepared	• Effects of all associated operations can be confirmed or cancelled; • information to retain confirm/cancel ability has been made persistent
decide to be prepared/cancel	• As "decide to be prepared"; • the persistent information specifies that the default action will be to cancel
decide to confirm autonomously	• Decision to confirm autonomously has been made persistent; • the effects of associated operations will be confirmed regardless of failures
decide to cancel autonomously	• Decision to cancel autonomously has been made persistent • the effects of associated operations will be cancelled regardless of failures
apply ordered confirmation	• Effects of all associated operations have been confirmed; • Persistent information is effectively removed
remove persistent information	• Persistent information is effectively removed;

detect problem	• For at least some of the associated operations, EITHER
	   o they cannot be consistently cancelled or consistently confirmed; OR
	   o it cannot be determined whether they will be cancelled or confirmed
	 • AND, information about this is not persistent

Event name	Meaning
detect and record problem	• As for the first condition of "detect problem" • information recording this has been persisted (to the degree considered appropriate), or the detection itself is persistent. (i.e. will be re-detected on recovery)

2541

2542

Table 3: Decision events for a Superior

Event name	Meaning
decide to request confirm	• All associated application messages to be sent to the service have been sent; • There are no other remaining Inferiors • All enrolments that would create other Inferiors have completed (no outstanding CONTEXT_REPLYs) • The Superior has been requested to confirm
decide to prepare	• All associated application messages to be sent to the service have been sent; • The Superior has been requested to prepare this Inferior

decide to confirm	• Either
	  o PREPARED or PREPARED/cancel has been received from all other remaining Inferiors; AND
	  o Superior has been requested to confirm; AND
	  o persistent information records the confirm decision and identifies all remaining Inferiors;
	• Or
	  o persistent information records an offer of confirmation and has been instructed to confirm

decide to cancel	• Superior has not offered confirmation; OR • Superior has offered confirmation and has been instructed to cancel; OR • Superior has offered confirmation but has made an autonomous cancellation decision
remove confirm information record contradiction	• Persistent information has been effectively removed; • Information recording the contradiction has been persisted (to the degree considered appropriate)

2543

2544   **Persistent information**

2545

2546 Persisted information (especially prepared information at an Inferior, confirm information at a
2547 Superior) may include qualifications of the state carried in Qualifiers of the corresponding
2548 message (e.g. inferior timeouts in prepared information). It may also include application-
2549 specific information (especially in Inferiors) to allow the future confirmation or cancellation
2550 of the associated operations. In some cases it will also include information allowing an
2551 application message sent with a BTP message (e.g. PREPARED) to be repeated.
2552
2553 The "effective" removal of persistent information allows for the possibility that the
2554 information is retained (perhaps for audit and tracing purposes) but some change to the
2555 persistent information (as a whole) means that if there is a failure after such change, on
2556 recovery, the persistent information does not cause the endpoint to return the state it would
2557 have recovered to before the change.
2558
2559 In all cases, the degree to which information described as "persistent" will survive failure is a
2560 configuration and implementation option. An implementation **should** describe the level of
2561 failure that it is capable of surviving. For applications manipulating information that is itself
2562 volatile (e.g. network configurations), there is no requirement to make the BTP state
2563 information more persistent that than the application information.
2564
2565 The degree of persistence of the recording of a hazard (problem) at an Inferior and recording
2566 of a detected contradiction at a Superior may be different from that applying to the persistent
2567 prepared and confirm information. Implementations and configuration may choose to pass
2568 hazard and contradiction information via management mechanisms rather than through BTP.
2569 Such passing of information to a management mechanism could be treated as "record
2570 problem" or "record contradiction".
2571

2572

Table 4 : Superior states

State	summary
I1	CONTEXT created
A1	ENROLing
B1	ENROLLED (active)
C1	resigning
D1	PREPARE sent
E1	PREPARED received
E2	PREPARED/cancel received
F1	CONFIRM sent
F2	completed after confirm
G1	cancel decided
G2	CANCEL sent
G3	cancelling, RESIGN received
G4	both cancelled
H1	inferior autonomously confirmed
J1	Inferior autonomously cancelled
K1	confirmed, contradiction detected
L1	cancelled, contradiction detected
P1	hazard reported
P2	hazard reported in null state
P3	hazard reported after confirm decision
P4	hazard reported after cancel decision
Q1	contradiction detected in null state
R1	Contradiction or hazard recorded
R2	completed after contradiction or hazard recorded
S1	REQUEST CONFIRM decided
Y1	completed queried
Z	completed and unknown

2573
2574

Table 5 : Inferior states

State	summary
i1	aware of CONTEXT
a1	enrolling
b1	enrolled
c1	resigning
d1	preparing
e1	prepared
e2	prepared,default to cancel
f1	confirming
f2	confirming after default cancel
g1	CANCEL received in prepared state
g2	CANCEL received in prepared/cancel state
h1	Autonomously confirmed
h2	autonomously confirmed, superior confirmed
j1	autonomously cancelled
j2	autonomously cancelled, superior cancelled
k1	autonomously cancelled, contradicted
k2	autonomously cancelled, CONTRADICTION received
l1	autonomously confirmed, contradicted
l2	autonomously confirmed, CONTRADICTION received
m1	confirmation applied
n1	cancelling
p1	hazard detected, not recorded
p2	hazard detected in prepared state, not recorded
q1	hazard recorded
s1	REQUEST CONFIRM received after prepared state
s2	REQUEST CONFIRM received
s3	REQUEST CONFIRM received, confirming
s4	REQUEST CONFIRM received, cancelling
s5	REQUEST CONFIRM received, hazard detected
s6	REQUEST CONFIRM received, hazard recorded
x1	completed, presuming abort
x2	completed, presuming abort after prepared/cancel
y1	completed, queried

State	summary
y2	completed, default cancel, a message received
z	completed
z1	completed with default cancel

2575
2576

Table 6: Superior state table – normal forward progression

	I1	A1	B1	C1	D1	E1	E2	F1	F2
`receive ENROL/rsp-req`	A1								
`receive ENROL/no-rsp-req`	B1								
`receive RESIGN/rsp-req`	Y1		C1	C1	C1				
`receive RESIGN/no-rsp-req`	Z		Z	Z	Z				
`receive PREPARED`	Y1		E1		E1	E1		F1	
`receive PREPARED/cancel`	Y1		E2		E2		E2	F1	
`receive CONFIRMED/auto`	Q1		H1		H1	H1		F1	
`receive CONFIRMED/response`								F2	F2
`receive CANCELLED`	Y1		Z		Z	J1	J1	K1	
`receive HAZARD`	P1	P1	P1		P1	P1	P1	P3	
`receive INF_STATE/active/y`	Y1	A1	B1		D1				
`receive INF_STATE/active`			B1		D1				
`receive INF_STATE/unknown`			Z	Z	Z				
`send ENROLLED`		B1							
`send RESIGNED`				Z					
`send PREPARE`					D1	E1	E2		
`send CONFIRM_ONE_PHASE`									
`send CONFIRM`								F1	
`send CANCEL`									
`send CONTRADICTION`									
`send SUP_STATE/active/y`			B1						
`send SUP_STATE/active`			B1						
`send SUP_STATE/prepared-rcvd/y`						E1	E2		
`send SUP_STATE/prepared-rcvd`						E1	E2		
`send SUP_STATE/unknown`									
`decide to request confirm`			S1			S1	S1		
`decide to prepare`			D1						
`decide to confirm`						F1	F1		
`decide to cancel`			G1		G1	G1	Z		
`remove persistent information`									Z
`record contradiction`									
`disruption I`	Z	Z	Z	Z	Z	Z	Z		F1
`disruption II`						D1	D1		
`disruption III`						B1	B1		
`disruption IV`									

2577 **Table 7: Superior state table – cancellation and contradiction**

	G1	G2	G3	G4	H1	J1	K1	L1
`receive ENROL/rsp-req`								
`receive ENROL/no-rsp-req`								
`receive RESIGN/rsp-req`	G3	Z	G3					
`receive RESIGN/no-rsp-req`	Z	Z	Z					
`receive PREPARED`	G1	G2						
`receive PREPARED/cancel`	G1	G2						
`receive CONFIRMED/auto`	L1	L1			H1			L1
`receive CONFIRMED/response`								
`receive CANCELLED`	G4	Z		G4		J1	K1	
`receive HAZARD`	P4	P4						
`receive INF_STATE/active/y`	G1	G2						
`receive INF_STATE/active`	G1	G2						
`receive INF_STATE/unknown`	Z	Z	Z	Z				
`send ENROLLED`								
`send RESIGNED`								
`send PREPARE`								
`send CONFIRM_ONE_PHASE`								
`send CONFIRM`								
`send CANCEL`	G2	G2	Z	Z				
`send CONTRADICTION`								
`send SUP_STATE/active/y`								
`send SUP_STATE/active`								
`send SUP_STATE/prepared-rcvd/y`								
`send SUP_STATE/prepared-rcvd`								
`send SUP_STATE/unknown`								
`decide to request confirm`								
`decide to prepare`								
`decide to confirm`					F1	K1		
`decide to cancel`					L1	G4		
`remove persistent information`								
`record contradiction`							R1	R1
`disruption I`	Z	Z	Z	Z	Z	Z	F1	Z
`disruption II`			G2	G2	E1	E1		G2
`disruption III`					D1	D1		
`disruption IV`					B1	B1		

2578

2578

Table 8: Superior state table – hazard and request confirm

	P1	P2	P3	P4	Q1	R1	R2	S1
receive ENROL/rsp-req								
receive ENROL/no-rsp-req								
receive RESIGN/rsp-req								C1
receive RESIGN/no-rsp-req								Z
receive PREPARED								S1
receive PREPARED/cancel								S1
receive CONFIRMED/auto					Q1	R1	R1	S1
receive CONFIRMED/response					Z	R2		Z
receive CANCELLED						R1	R1	Z
receive HAZARD	P1	P2	P3	P4		R1	R1	Z
receive INF_STATE/active/y								S1
receive INF_STATE/active								S1
receive INF_STATE/unknown	P1	P2		P4		R2	R2	Z
send ENROLLED								
send RESIGNED								
send PREPARE								
send CONFIRM_ONE_PHASE								S1
send CONFIRM								
send CANCEL								
send CONTRADICTION						R2		
send SUP_STATE/active/y								
send SUP_STATE/active								
send SUP_STATE/prepared-rcvd/y								
send SUP_STATE/prepared-rcvd								
send SUP_STATE/unknown								
decide to request confirm								
decide to prepare								
decide to confirm								
decide to cancel								
remove persistent information							Z	
record contradiction	R1	R1	R1	R1	R1			
disruption I	Z	Z	Z	Z	Z		R1	Z
disruption II	D1		F1	G2				
disruption III	B1							
disruption IV								

2579

2579 **Table 9: Superior state table – query after completion and completed states**

	Y1	Z
receive ENROL/rsp-req		Y1
receive ENROL/no-rsp-req		Y1
receive RESIGN/rsp-req	Y1	Y1
receive RESIGN/no-rsp-req	Z	Z
receive PREPARED	Y1	Y1
receive PREPARED/cancel	Y1	Y1
receive CONFIRMED/auto	Q1	Q1
receive CONFIRMED/response	Z	Z
receive CANCELLED	Y1	Y1
receive HAZARD	P2	P2
receive INF_STATE/active/y	Y1	Y1
receive INF_STATE/active	Y1	Z
receive INF_STATE/unknown	Z	Z
send ENROLLED		
send RESIGNED		
send PREPARE		
send CONFIRM_ONE_PHASE		
send CONFIRM		
send CANCEL		
send CONTRADICTION		
send SUP_STATE/active/y		
send SUP_STATE/active		
send SUP_STATE/prepared-rcvd/y		
send SUP_STATE/prepared-rcvd		
send SUP_STATE/unknown	Z	
decide to request confirm		
decide to prepare		
decide to confirm		
decide to cancel		
remove persistent information		
record contradiction		
disruption I	Z	
disruption II		
disruption III		
disruption IV		

2580
2581

Table 10: Inferior state table – normal forward progression

	i1	a1	b1	c1	d1	e1	e2	f1	f2
send ENROL/rsp-req	a1								
send ENROL/no-rsp-req	b1								
send RESIGN/rsp-req				c1					
send RESIGN/no-rsp-req				z					
send PREPARED						e1			
send PREPARED/cancel							e2		
send CONFIRMED/auto									
send CONFIRMED/response									
send CANCELLED			z		z				
send HAZARD									
send INF_STATE/active/y		a1	b1		d1				
send INF_STATE/active			b1		d1				
send INF_STATE/unknown									
receive ENROLLED		b1							
receive RESIGNED				z					
receive PREPARE		d1	d1	c1	d1	e1	e2		
receive CONFIRM_ONE_PHASE		s2	s2	c1		s1	s1		
receive CONFIRM						f1	f2	f1	f2
receive CANCEL		n1	n1	z	n1	g1	g2		
receive CONTRADICTION									
receive SUP_STATE/active/y		b1	b1	c1		e1	e2		
receive SUP_STATE/active		b1	b1	c1		e1	e2		
receive SUP_STATE/prepared-rcvd/y						e1	e2		
receive SUP_STATE/prepared-rcvd						e1	e2		
receive SUP_STATE/unknown		z	z	z	z	x1	x2		
decide to resign			c1		c1				
decide to be prepared			e1		e1				
decide to be prepared/cancel			e2		e2				
decide to confirm autonomously						h1			
decide to cancel autonomously						j1	z1		
apply ordered confirmation								m1	m1
remove persistent information									
detect problem		p1	p1		p1	p2	p2	p2	p2
detect and record problem									
disruption I		z	z	z	z			e1	e2
disruption II					b1				
disruption III									

2584

Table 11: Inferior state table – cancellation and contradiction

	g1	g2	h1	h2	j1	j2	k1	k2	l1	l2
send ENROL/rsp-req										
send ENROL/no-rsp-req										
send RESIGN/rsp-req										
send RESIGN/no-rsp-req										
send PREPARED										
send PREPARED/cancel										
send CONFIRMED/auto			h1						l1	
send CONFIRMED/response										
send CANCELLED					j1		k1			
send HAZARD										
send INF_STATE/active/y										
send INF_STATE/active										
send INF_STATE/unknown										
receive ENROLLED										
receive RESIGNED										
receive PREPARE			h1		j1					
receive CONFIRM_ONE_PHASE			s3		s4					
receive CONFIRM			h2	h2	k1		k1			
receive CANCEL	g1	g2	l1		j2	j2			l1	
receive CONTRADICTION			l2		k2		k2	k2	l2	l2
receive SUP_STATE/active/y			h1		j1					
receive SUP_STATE/active			h1		j1					
receive SUP_STATE/prepared-rcvd/y			h1		j1					
receive SUP_STATE/prepared-rcvd			h1		j1					
receive SUP_STATE/unknown	x1	x2	l1		j2	j2	k2	k2	l1	
decide to resign										
decide to be prepared										
decide to be prepared/cancel										
decide to confirm autonomously										
decide to cancel autonomously										
apply ordered confirmation										
remove persistent information	n1	n1		m1		z		z		z
detect problem	p2	p2								
detect and record problem										
disruption I	e1	e2		h1		j1	j1	k1	h1	l1
disruption II								j1		h1
disruption III										

2585
2586

2586 **Table 12: Inferior state table– confirm, cancel ordered and hazard recording**

	m1	n1	p1	p2	q1
send ENROL/rsp-req					
send ENROL/no-rsp-req					
send RESIGN/rsp-req					
send RESIGN/no-rsp-req					
send PREPARED					
send PREPARED/cancel					
send CONFIRMED/auto					
send CONFIRMED/response	z				
send CANCELLED		z			
send HAZARD			p1	p2	q1
send INF_STATE/active/y					
send INF_STATE/active					
send INF_STATE/unknown					
receive ENROLLED			p1		q1
receive RESIGNED					
receive PREPARE			p1	p2	q1
receive CONFIRM_ONE_PHASE			s5	s5	s6
receive CONFIRM	m1			p2	q1
receive CANCEL		n1	p1	p2	q1
receive CONTRADICTION			z	z	z
receive SUP_STATE/active/y			p1	p2	q1
receive SUP_STATE/active			p1	p2	q1
receive SUP_STATE/prepared-rcvd/y				p2	q1
receive SUP_STATE/prepared-rcvd				p2	q1
receive SUP_STATE/unknown		z	p1	p2	q1
decide to resign					
decide to be prepared					
decide to be prepared/cancel					
decide to confirm autonomously					
decide to cancel autonomously					
apply ordered confirmation					
remove persistent information					
detect problem					
detect and record problem			q1	q1	
disruption I	z	z	z		
disruption II		d1			
disruption III		b1			

2587
2588

2588

Table 13: Inferior state table – request confirm states

	s1	s2	s3	s4	s5	s6
send ENROL/rsp-req						
send ENROL/no-rsp-req						
send RESIGN/rsp-req						
send RESIGN/no-rsp-req						
send PREPARED						
send PREPARED/cancel						
send CONFIRMED/auto						
send CONFIRMED/response			z			
send CANCELLED				z		
send HAZARD					z	z
send INF_STATE/active/y						
send INF_STATE/active						
send INF_STATE/unknown						
receive ENROLLED						
receive RESIGNED						
receive PREPARE						
receive CONFIRM_ONE_PHASE	s1	s2	s3	s4	s5	s6
receive CONFIRM						
receive CANCEL						
receive CONTRADICTION			s3		z	s6
receive SUP_STATE/active/y						
receive SUP_STATE/active						
receive SUP_STATE/prepared-rcvd/y						
receive SUP_STATE/prepared-rcvd						
receive SUP_STATE/unknown	x1	z	z	z	z	z
decide to resign						
decide to be prepared						
decide to be prepared/cancel						
decide to confirm autonomously		s3				
decide to cancel autonomously		s4				
apply ordered confirmation						
remove persistent information	s2					
detect problem						
detect and record problem		s6				
disruption I	e1	z		z	z	
disruption II						
disruption III						

2589

2589 **Table 14: Inferior state table– completed states (including presume -abort and queried)**

	x1	x2	y1	y2	z	z1
send ENROL/rsp-req						
send ENROL/no-rsp-req						
send RESIGN/rsp-req						
send RESIGN/no-rsp-req						
send PREPARED						
send PREPARED/cancel						
send CONFIRMED/auto						
send CONFIRMED/response						
send CANCELLED				z1		
send HAZARD						
send INF_STATE/active/y						
send INF_STATE/active						
send INF_STATE/unknown			z			
receive ENROLLED					z	
receive RESIGNED			y1		z	
receive PREPARE			y1	y2	y1	z1
receive CONFIRM_ONE_PHASE			y1	y2	y1	y1
receive CONFIRM				y2	m1	y2
receive CANCEL			y1	z	y1	y1
receive CONTRADICTION			z	z	z	z
receive SUP_STATE/active/y			y1	y2	y1	y2
receive SUP_STATE/active			y1	y2	z	z1
receive SUP_STATE/prepared-rcvd/y				y2		y2
receive SUP_STATE/prepared-rcvd				y2		y2
receive SUP_STATE/unknown	x1	x2	y1	y2	z	z
decide to resign						
decide to be prepared						
decide to be prepared/cancel						
decide to confirm autonomously						
decide to cancel autonomously						
apply ordered confirmation						
remove persistent information	z	z				
detect problem						
detect and record problem						
disruption I	e1	e2				
disruption II						
disruption III						

2590
2591
2592

Failure Recovery

Types of failure

BTP is designed to ensure the delivery of a consistent decision for a business transaction to the parties involved, even in the event of failure. Failures can be classified as:

> **Communication failure**: messages between BTP actors are lost and not delivered. BTP assumes the carrier protocol ensures that messages are either delivered correctly (without corruption) or are lost, but does not assume that all losses are reported or that messages sent separately are delivered in the order of sending.

> **Node failure (system failure, site failure**): a machine hosting one or more BTP actors stops processing and all its volatile data is lost. BTP assumes a site fails by stopping – it either operates correctly or not at all, it never operates incorrectly.

Communication failure may become known to a BTP implementation by an indication from the lower layers or may be inferred (or suspected) by the expiry of a timeout. Recovery from a communication failure requires only that the two actors can again send messages to each other and continue or complete the progress of the business transaction. In the state tables for the Superior:Inferior relationship, each side is either waiting to make a decision or can send a message. For some states, the message to be sent is a repetition of a regular message; for other states, the INFERIOR_STATE or SUPERIOR_STATE message can be sent, requesting a response. Thus, following a communication failure, either side can prompt the other to re-establish the relationship. Receiving one of the *_STATE messages asking for a response does not require an immediate response – especially if an implementation is waiting to determine a decision (perhaps because it is itself waiting for a decision from elsewhere), an implementation may choose not to reply until it wishes too.

A node failure is distinguished from communication failure because there is loss of volatile state. To ensure consistent application of the decision of a business transaction, BTP requires that some state information will be persisted despite node failure. Exactly what real events correspond to node failure but leave the persistent information undamaged is a matter for implementation choice, depending on application requirements; however, for most application uses, power failure should be survivable (an exception would be if the data manipulated by the associated operations was volatile). There will always be some level of event sufficiently catastrophic to lose persistent information and the ability to recover– destruction of the computer or bankruptcy of the organisation, for example.

Recovery from node failure involves recreating the endpoint in a node that has access to the persistent information for incomplete transactions. This may be a recreation of the original node (including the ability to perform application work) using the same addresses; or there may be a distinct recovery entity, which can access the persistent data, but has a different address; other implementation approaches are possible. Restoration of the endpoint from persistent information will often result in a partial loss of state, relative to the volatile state reached before the failure. This is modelled in the state tables by the "disruption" events. After recovery from node failure, the implementation behaves much as if a communication failure had occurred.

Persistent information

2643   BTP requires that some decision events are persisted – that information recording an
2644   Inferior's decision to be prepared, a Superior's decision to confirm and an Inferior's
2645   autonomous decision survive failure. Making the first two decisions persistent ensures that a
2646   consistent decision can be reached for the business transaction and that it is delivered to all
2647   involved nodes. Requiring an Inferior's autonomous decision to be persistent allows BTP to
2648   ensure that, if this decision is contradictory (i.e. opposite to the decision at the Superior), the
2649   contradiction will be reported to the Superior, despite failures.

2651   BTP also permits, but does not require, recovery of the Superior:Inferior relationship in the
2652   active state (unlike many transaction protocols, where a communication or endpoint failure in
2653   active state would invariably cause rollback of the transaction). Recovery in the active state
2654   may require that the application exchange is resynchronised as well – BTP does not directly
2655   support this, but does allow continuation of the business transaction as such. In the state
2656   tables, from some states, there are several levels of disruption, distinguished by which state
2657   the implementation transits to – this represents the survival of different extents of state
2658   information over failure and recovery. The different levels of disruption describe legitimate
2659   states for the endpoint to be in after it has recovered – **they do not require that all**
2660   **implementations are able to exhibit the appropriate partial loss of state information**.
2661   The absence of a destination state for the disruption events means that such a transition is not
2662   legitimate – thus, for example, an Inferior that has decided to be prepared will always recover
2663   to the same state, by virtue of the information persisted in the "decide to be prepared" event.

2665   Apart from the (optional) recovery in active state, BTP follows the well-known presume-
2666   abort model – it is only required that information be persisted when decisions are made (and
2667   not, e.g. on enrolment). This means that on recovery, one side may have persistent
2668   information but the other does not. This occurs when an Inferior has decided to be prepared
2669   but the Superior never confirmed (so the decision is "presumed" to be cancel), or because the
2670   Superior did confirm, and the Inferior applied the confirm, removed its persistent information
2671   but the acknowledgement (CONFIRMED) was never received by the Superior (or, at least, it
2672   still had the persistent information when the failure occurred).

2674   Information to be persisted for an Inferior's "decision to be prepared" must be sufficient to
2675   re-establish communication with the Superior, to apply a confirm decision and to apply a
2676   cancel decision. It will thus need to include
2677   Inferior identity (this may be an index used to locate the information)
2678   Superior address (as on CONTEXT)
2679   Superior identifier (as on CONTEXT)
2680   default-is-cancel value (as on PREPARED)

2682   The information needed to apply confirm/cancel decisions will depend on the application and
2683   the associated operations. It may also normally be necessary to persist any qualifiers that
2684   were sent with the PREPARED message or application messages sent with the PREPARED,
2685   since the PREPARED message will be repeated if a failure occurs.

2687 A Superior must record corresponding information to allow it to re-establish communication
2688 with the Inferior:
2689   Inferior address (as on ENROL)
2690   Inferior identifier (as on ENROL)
2691
2692 A Superior that is the Decider for the business transaction need only persist this information
2693 if it makes a decision to confirm (and this Inferior is in the confirm set, for a Cohesion). A
2694 Superior that is also an Inferior to some other entity (i.e. it is an intermediate in a tree, as
2695 atom in a cohesion, sub-coordinator or sub-composer) must persist this information as
2696 Superior (to this Inferior) as part of the persistent information of its decision to be prepared
2697 (as an Inferior). For such an entity, the "decision to confirm" as Superior is made when (and
2698 if) CONFIRM is received from its Superior or it makes an autonomous decision to confirm. If
2699 CONFIRM is received, the persistent information may be changed to show the confirm
2700 decision, but alternatively, the receipt of the CONFIRM can be treated as the decision itself.
2701 If the persistent information is left unchanged and there is a node failure, on recovery the
2702 entity (as an Inferior) will be in a prepared state, and will rediscover the confirm decision
2703 (using the recovery exchanges to its Superior) before propagating it to its Inferior(s).
2704
2705 After failure, an implementation may not be able to restore an endpoint to the appropriate
2706 state immediately – in particular, the necessary persistent information may be inaccessible,
2707 although the implementation can respond to received BTP messages. In such a case, a
2708 Superior may reply to any BTP message except INFERIOR_STATE/* (i.e. with a "reply-
2709 requested" value "false") with SUPERIOR_STATE/inaccessible and an Inferior to any BTP
2710 message except SUPERIOR_STATE/* with "INFERIOR_STATE/inaccessible. Receipt of
2711 the *_STATE/inaccessible messages has no effect on the endpoint state.
2712
2713 **Redirection**
2714
2715 As described above, BTP uses the presume-abort model for recovery. A corollary of this is
2716 that there are cases where one side will attempt to re-establish communication when there is
2717 no persistent information for the relationship at the far-end. In such cases, it is important the
2718 side that is attempting recovery can distinguish between unsuccessful attempts to connect to
2719 the holder of the persistent information and when the information no longer exists. If the peer
2720 information does not exist, this side can draw conclusions and complete appropriately; if they
2721 merely fail to get through they are stuck in attempting recovery.
2722
2723 Two mechanisms are provided to make it possible that even when one side of a
2724 Superior:Inferior relationship has completed, that a message can eventually get through to
2725 something that can definitively report the status, distinguishing this case from a temporary
2726 inability to access the state of a continuing transaction element. The mechanisms are:
2727   o   Address fields which provide a "callback address" can be a set of addresses,
2728   which are alternatives one of which is chosen as the target address for the
2729   future message. If the sender of that message finds the address does not work,
2730   it can try a different alternative.
2731   o   The REDIRECT message can be used to inform the peer that an address
2732   previously given is no longer valid and to supply a replacement address (or
2733   set of addresses). REDIRECT can be issued either as a response to receipt of
2734   a message or spontaneously.

2735

2736 The two mechanisms can be used in combination, with one or more of the original set of
2737 addresses just being a redirector, which does not itself ever have direct access to the state
2738 information for the transaction, but will respond to any message with an appropriate
2739 REDIRECT.
2740

2741 An alternative implementation approach is to have a single addressable entity that uses the
2742 same address for all transactions, distinguishing them by identifier, and which always
2743 recovers to use the same address. Such an implementation would not need to supply
2744 "backup" addresses (and would only use REDIRECT if it was being permanently migrated).

2745 ### Terminator:Decider failures

2746

2747 BTP does not provide facilities or impose requirements on the recovery of
2748 Terminator:Decider relationships, other than allowing messages to be repeated. A Terminator
2749 may survive failures (by retaining knowledge of the Decider's address and identifier), but this
2750 is an implementation option. Although a Decider (if it decides to confirm) will persist
2751 information about the confirm decision, it is not required, after failure, to remain accessible
2752 using the inferior address it offered to the Terminator. Any such recovery is an
2753 implementation option.
2754

2755 A Decider's address (as returned on BEGUN) may be a set of addresses, allowing a failed
2756 Decider to be recovered at a different address.
2757

2758 A Decider has no way of initiating a call to a Terminator to ensure that it is still active, and
2759 thus no way of detecting that a Terminator has failed. To avoid a Decider waiting for ever for
2760 a REQUEST_CONFIRM that will never arrive, the standard qualifier "Transaction timelimit"
2761 can be used (by the Initiator) to inform the Decider when it can assume the Terminator will
2762 not issue REQUEST_CONFIRM and so it (the Decider) should initiate cancellation.
2763

2764 ## XML representation of Message Set

2765

2766 This section describes the syntax for BTP messages in XML. These XML messages represent
2767 a midpoint between the abstract messages and what actually gets sent on the wire.
2768

2769 All BTP related URIs have been created using Oasis URI conventions as specified in RFC
2770 3121
2771

2772 The XML Namespace for the BTP messages is urn:oasis:names:tc:BTP:xml
2773

2774 In addition to an XML schema, this specification uses an informal syntax to describe the
2775 structure of the BTP messages. The syntax appears as an XML instance, but the values
2776 contain data types instead of values. The following symbols are appended to some of the
2777 XML constructs: ? (zero or one), * (zero or more), + (one or more.) The absence of one of
2778 these symbols corresponds to "one and only one."
2779

2780 ### Addresses

2781

2782 As described in the "Abstract Message and Associated Contracts – Addresses" section, a BTP
2783 address comprises three parts, and for a target address only the "additional information" field
2784 is inside the BTP messages. For all BTP messages whose abstract form includes a target
2785 address parameter, the corresponding XML representation includes a "target-additional-
2786 information" element. This element may be omitted if it would be empty.
2787
2788 For other addresses, all three fields are represent, as in:
2789
```
2790     <btp:some-address>
2791       <btp:binding-name>...carrier binding URI...</btp:binding-name>
2792       <btp:binding-address>...carrier specific URI...</btp:binding-
2793     address>
2794       <btp:additional-information>...optional additional addressing
2795     information...</btp:additional-information> ?
2796     </btp:some-address>
```
2797
2798
2799 A "published" address can be a set of <some-address>, which are alternatives which can be
2800 chosen by the peer (sender.) Multiple addresses are used in two cases: different bindings to
2801 same endpoint, or backup endpoints. In the former, the receiver of the message has the choice
2802 of which address to use (depending on which binding is preferable.) In the case where
2803 multiple addresses are used for redundancy, a **priority** attribute can be specified to help the
2804 receiver choose among the addresses- the address with the highest priority should be used,
2805 other things being equal. The **priority** is used as a hint and does not enforce any behaviour in
2806 the receiver of the message. Default priority is a value of 1.
2807

2808 ## Qualifiers
2809 The "Qualifier name" is used as the element name, within the namespace of the "Qualifier
2810 group".
2811

2812 ## Examples:
```
2813     <btpq:inferior-timeout
2814         xmlns:btpq="urn:oasis:names:tc:BTP:qualifiers"
2815         xmlns:btp="urn:oasis:names:tc:BTP:xml"
2816         btp:must-be-understood="false"
2817         btp:to-be-propagated="false">1800</auth:username>
2818
2819     <auth:username
2820         xmlns:auth="http://www.example.com/ns/auth"
2821         xmlns:btp="urn:oasis:names:tc:BTP:xml"
2822         btp:must-be-understood="true"
2823         btp:to-be-propagated="true">jtauber</auth:username>
```
2824
2825 Attributes **must-be-understood** has default value "true" and **to-be-propagated** has default
2826 value "false".
2827

2828 ## Identifiers
2829 Unspecified length strings made of up hexadecimal digits (0->9, A->F). Note: lower case a->f
2830 are not valid.
2831

2832 Examples: "01", "FAB224234CCCC2"
2833
2834 Note – Use of hexadecimal digits avoids problems with character-code representations. The
2835 only operation the BTP implementations have to perform on identifiers is to match them.
2836
2837 Message References
2838 Each BTP message has an optional **id** attribute to give it a unique identifier. An application
2839 can make use of those identifiers, but no processing is enforced.
2840

Messages

2842

CONTEXT

2844

```
<btp:context id? superior-type="cohesion|atom">
  <btp:superior-address>  +
     ...address...
  </btp:superior-address>
  <btp:superior-identifier>...hexstring...</btp:superior-
identifier>
  <btp:qualifiers> ?
     ...qualifiers...
  </btp:qualifiers>
</btp:context>
```

2855
2856

CONTEXT-REPLY

2858

```
<btp:context-reply id? superior-type="cohesion|atom">
  <btp:superior-address>  +
          ...address...
  </btp:superior-address>
  <btp:superior-identifier>...hexstring...</btp:superior-
identifier>
  <completion-status>completed|related|repudiated</completion-
status>
  <btp:qualifiers> ?
     ...qualifiers...
  </btp:qualifiers>
</btp:context>
```

2871
2872

BEGIN

2874

```
<btp:begin id? transaction-type="cohesion|atom">
  <btp:target-additional-information>
     ...additional address information...
  </btp:target-additional-information>
  <btp:reply-address>
     ...address...
  </btp:reply-address>
  <btp:qualifiers> ?
     ...qualifiers...
```

```
2884          </btp:qualifiers>
2885        </btp:begin>
2886
2887
2888    BEGUN
2889
2890        <btp:begun id? transaction-type="cohesion|atom">
2891          <btp:target-additional-information>
2892            ...additional address information...
2893          </btp:target-additional-information>
2894          <btp:decider-address> ?
2895            ...address...
2896          </btp:decider-address>
2897          <btp:transaction-identifier>...hexstring...</btp:transaction-
2898        identifier> ?
2899          <btp:inferior-handle>...hexstring...</btp:inferior:handle> ?
2900          <btp:inferior-address> ?
2901            ...address...
2902          </btp:inferior-address>
2903          <btp:qualifiers> ?
2904            ...qualifiers...
2905          </btp:qualifiers>
2906        </btp:begun>
2907
2908
2909    ENROL
2910
2911        <btp:enrol reply-requested="true|false" id?>
2912          <btp:target-additional-information>
2913            ...additional address information...
2914          </btp:target-additional-information>
2915          <btp:superior-identifier>...hexstring...</btp:superior-
2916        identifier>
2917          <btp:reply-address>  ?
2918            ...address...
2919          </btp:reply-address>
2920          <btp:inferior-address>  +
2921            ...address...
2922          </btp:inferior-address>
2923          <btp:inferior-identifier>...hexstring...</btp:inferior-
2924        identifier>
2925          <btp:qualifiers> ?
2926            ...qualifiers...
2927          </btp:qualifiers>
2928        </btp:enrol>
2929
2930
2931    ENROLLED
2932
2933        <btp:enrolled id?>
2934        <btp:target-additional-information>
2935            ...additional address information...
2936          </btp:target-additional-information>
```

```
    <btp:inferior-identifier>...hexstring...</btp:inferior-
identifier>
    <btp:inferior-handle>...hexstring...</btp:inferior:handle> ?
    <btp:qualifiers> ?
       ...qualifiers...
    </btp:qualifiers>
</btp:enrolled>
```

RESIGN

```
<btp:resign response-requested="true|false" id?>
<btp:target-additional-information>
    ...additional address information...
  </btp:target-additional-information>
  <btp:superior-identifier>...hexstring...</btp:superior-
identifier>
  <btp:inferior-address>  +
    ...address...
  </btp:inferior-address>
  <btp:inferior-identifier>...hexstring...</btp:inferior-
identifier>
  <btp:qualifiers> ?
    ...qualifiers...
  </btp:qualifiers>
</btp:resign>
```

RESIGNED

```
<btp:resigned id?>
  <btp:target-additional-information>
    ...additional address information...
  </btp:target-additional-information>
  <btp:inferior-identifier>...hexstring...</btp:inferior-
identifier>
  <btp:qualifiers> ?
    ...qualifiers...
  </btp:qualifiers>
</btp:resigned>
```

PREPARE

```
<btp:prepare id?>
  <btp:target-additional-information>
    ...additional address information...
  </btp:target-additional-information>
  <btp:inferior-identifier>...hexstring...</btp:inferior-
identifier> ?
  <btp:reply-address>  ?
    ...address...
  </btp:reply-address>
```

```
2990        <btp:transaction-identifier>...hexstring...</btp:transaction-
2991   identifier> ?
2992        <btp:inferiors-list> ?
2993            <btp:inferior-handle>...hexstring...</btp:inferior-handle>
2994   +
2995        </btp:inferiors-list>
2996        <btp:qualifiers> ?
2997          ...qualifiers...
2998        </btp:qualifiers>
2999      </btp:prepare>
3000
3001
```

PREPARED

```
3004      <btp:prepared default-is-cancel="false|true" id?>
3005        <btp:target-additional-information>
3006          ...additional address information...
3007        </btp:target-additional-information>
3008        <btp:superior-identifier>...hexstring...</btp:superior-
3009   identifier>
3010        <btp:inferior-address> +
3011          ...address...
3012        </btp:inferior-address>
3013        <btp:inferior-identifier>...hexstring...</btp:inferior-
3014   identifier>
3015        <btp:qualifiers> ?
3016          ...qualifiers...
3017        </btp:qualifiers>
3018      </btp:prepared>
3019
3020
```

CONFIRM

```
3023      <btp:confirm id?>
3024        <btp:target-additional-information>
3025          ...additional address information...
3026        </btp:target-additional-information>
3027        <btp:inferior-identifier>...hexstring...</btp:inferior-
3028   identifier>
3029        <btp:qualifiers> ?
3030          ...qualifiers...
3031        </btp:qualifiers>
3032      </btp:confirm>
3033
3034
```

CONFIRMED

```
3037      <btp:confirmed confirmed-received="true|false" id?>
3038        <btp:target-additional-information>
3039          ...additional address information...
3040        </btp:target-additional-information>
3041        <btp:superior-identifier>...hexstring...</btp:superior-
3042   identifier>
```

```
3043        <btp:inferior-address> ?
3044          ...address...
3045        </btp:inferior-address>
3046        <btp:inferior-identifier>...hexstring...</btp:inferior-
3047     identifier> ?
3048        <btp:decider-address> ?
3049          ...address...
3050        </btp:decider-address>
3051        <btp:transaction-identifier>...hexstring...</btp:transaction-
3052     identifier> ?
3053        <btp:qualifiers> ?
3054          ...qualifiers...
3055        </btp:qualifiers>
3056      </btp:confirmed>
3057
3058
3059     CANCEL
3060
3061      <btp:cancel id?>
3062        <btp:target-additional-information>
3063          ...additional address information...
3064        </btp:target-additional-information>
3065        <btp:inferior-identifier>...hexstring...</btp:inferior-
3066     identifier> ?
3067        <btp:reply-address>  ?
3068          ...address...
3069        </btp:reply-address>
3070        <btp:transaction-identifier>...hexstring...</btp:transaction-
3071     identifier> ?
3072        <btp:inferiors-list> ?
3073            <btp:inferior-handle>...hexstring...</btp:inferior-handle>
3074        </btp:inferiors-list>
3075        <btp:qualifiers> ?
3076          ...qualifiers...
3077        </btp:qualifiers>
3078      </btp:cancel>
3079
3080
3081     CANCELLED
3082
3083      <btp:cancelled id?>
3084        <btp:target-additional-information>
3085          ...additional address information...
3086        </btp:target-additional-information>
3087        <btp:superior-identifier>...hexstring...</btp:superior-
3088     identifier>
3089        <btp:inferior-address> +
3090          ...address...
3091        </btp:inferior-address> ?
3092        <btp:inferior-identifier>...hexstring...</btp:inferior-
3093     identifier> ?
3094        <btp:decider-address> ?
3095          ...address...
3096        </btp:decider-address>
```

```
3097    <btp:transaction-identifier>...hexstring...</btp:transaction-
3098    identifier> ?
3099      <btp:qualifiers> ?
3100        ...qualifiers...
3101      </btp:qualifiers>
3102    </btp:cancelled>
3103
3104
```

HAZARD

```
3107    <btp:hazard id?>
3108      <btp:target-additional-information>
3109        ...additional address information...
3110      </btp:target-additional-information>
3111      <btp:superior-identifier>...hexstring...</btp:superior-
3112    identifier>
3113      <btp:inferior-address> +
3114        ...address...
3115      </btp:inferior-address>
3116      <btp:inferior-identifier>...hexstring...</btp:inferior-
3117    identifier>
3118      <btp:qualifiers> ?
3119        ...qualifiers...
3120      </btp:qualifiers>
3121    </btp:hazard>
3122
3123
```

CONTRADICTION

```
3126    <btp:contradiction id?>
3127      <btp:target-additional-information>
3128        ...additional address information...
3129      </btp:target-additional-information>
3130      <btp:inferior-identifier>...hexstring...</btp:inferior-
3131    identifier>
3132      <btp:qualifiers> ?
3133        ...qualifiers...
3134      </btp:qualifiers>
3135    </btp:contradiction>
3136
3137
```

SUPERIOR_STATE

```
3140    <btp:superior-state reply-requested="true|false" id?>
3141      <btp:target-additional-information>
3142        ...additional address information...
3143      </btp:target-additional-information>
3144      <btp:inferior-identifier>...hexstring...</btp:inferior-
3145    identifier>
3146      <btp:status>active|prepared-
3147    received|inaccessible|unknown</btp:status>
3148      <btp:qualifiers> ?
3149        ...qualifiers...
```

```
3150          </btp:qualifiers>
3151        </btp:superior-state>
3152
3153
3154
3155    INFERIOR_STATE
3156
3157        <btp:inferior-state reply-requested="true|false" id?>
3158          <btp:target-additional-information>
3159            ...additional address information...
3160          </btp:target-additional-information>
3161          <btp:superior-identifier>...hexstring...</btp:superior-
3162    identifier>
3163          <btp:inferior-address> +
3164            ...address...
3165          </btp:inferior-address>
3166          <btp:inferior-identifier>...hexstring...</btp:inferior-
3167    identifier>
3168          <btp:status> active|prepared-
3169    received|inaccessible|unknown</btp:status>
3170          <btp:qualifiers> ?
3171            ...qualifiers...
3172          </btp:qualifiers>
3173        </btp:inferior-state>
3174
3175
3176
3177    CONFIRM_ONE_PHASE
3178
3179        <btp:confirm-one-phase report-hazard="true|false" id?>
3180          <btp:target-additional-information>
3181            ...additional address information...
3182          </btp:target-additional-information>
3183          <btp:inferior-identifier>...hexstring...</btp:inferior-
3184    identifier>
3185          <btp:qualifiers> ?
3186            ...qualifiers...
3187          </btp:qualifiers>
3188        </btp:confirm-one-phase>
3189
3190
3191    REQUEST_CONFIRM
3192
3193        <btp:request_confirm report-hazard="true|false" id?>
3194          <btp:target-additional-information>
3195            ...additional address information...
3196          </btp:target-additional-information>
3197          <btp:reply-address>
3198            ...address...
3199          </btp:reply-address>
3200          <btp:transaction-identifier>...hexstring...</btp:transaction-
3201    identifier>
3202          <btp:inferiors-list> ?
```

```
3203           <btp:inferior-handle>...hexstring...</btp:inferior-handle>
3204    +
3205      </btp:inferiors-list>
3206      <btp:qualifiers> ?
3207        ...qualifiers...
3208      </btp:qualifiers>
3209    </btp:request_confirm>
3210
3211
```

REQUEST_STATUSES

```
3214    <btp:request_statuses id?>
3215      <btp:target-additional-information>
3216        ...additional address information...
3217      </btp:target-additional-information>
3218      <btp:reply-address>
3219        ...address...
3220      </btp:reply-address>
3221      <btp:transaction-identifier>...hexstring...</btp:transaction-
3222    identifier>
3223      <btp:inferiors-list> ?
3224           <btp:inferior-handle>...hexstring...</btp:inferior-handle>
3225    +
3226      </btp:inferiors-list>
3227      <btp:qualifiers> ?
3228        ...qualifiers...
3229      </btp:qualifiers>
3230    </btp:request_statuses>
3231
3232
```

INFERIOR_STATUSES

```
3235    <btp:inferior_statuses id?>
3236      <btp:target-additional-information>
3237        ...additional address information...
3238      </btp:target-additional-information>
3239      <btp:decider-address>
3240        ...address...
3241      </btp:decider-address>
3242      <btp:transaction-identifier>...hexstring...</btp:transaction-
3243    identifier>
3244      <btp:status-list>
3245           <btp:status-item> +
3246               <btp:inferior-handle>...hexstring...</btp:inferior-
3247    handle>
3248               <btp:status>active|resigned|preparing|prepared|
3249                   autonomously-confirmed|autonomously-cancelled|
3250                   confirming|confirmed|cancelling|cancelled|
3251                   cancel-contradiction|confirm-contradiction|
3252                   hazard</btp:status>
3253               <btp:qualifiers> ?
3254                   ...qualifiers...
3255               </btp:qualifiers>
3256           </btp:status-item>
```

```
3257        </btp:status-list>
3258        <btp:qualifiers> ?
3259          ...qualifiers...
3260        </btp:qualifiers>
3261      </btp:inferior_statuses>
3262
3263
```

REQUEST_STATUS

```
3266      <btp:request_status id?>
3267        <btp:target-additional-information>
3268          ...additional address information...
3269        </btp:target-additional-information>
3270        <btp:reply-address>
3271          ...address...
3272        </btp:reply-address>
3273        <btp:inferior-identifier>...hexstring...</btp:inferior-
3274      identifier> ?
3275        <btp:transaction-identifier>...hexstring...</btp:transaction-
3276      identifier> ?
3277        <btp:qualifiers> ?
3278          ...qualifiers...
3279        </btp:qualifiers>
3280      </btp:request_status>
3281
3282
```

STATUS

```
3285      <btp:status id?>
3286        <btp:target-additional-information>
3287          ...additional address information...
3288        </btp:target-additional-information>
3289        <btp:inferior-address> ?
3290          ...address...
3291        </btp:inferior-address>
3292        <btp:inferior-identifier>...hexstring...</btp:inferior-
3293      identifier> ?
3294        <btp:decider-address> ?
3295          ...address...
3296        </btp:decider-address>
3297        <btp:transaction-identifier>...hexstring...</btp:transaction-
3298      identifier> ?
3299        <btp:status-value>created|enrolling|active|resigning|
3300              resigned|preparing|prepared|
3301              confirming|confirmed|cancelling|cancelled|
3302              cancel-contradiction|confirm-contradiction|
3303              hazard|contradicted|unknown|inaccessible</btp:status-
3304      value>
3305        <btp:qualifiers> ?
3306          ...qualifiers...
3307        </btp:qualifiers>
3308      </btp:status>
3309
```

REDIRECT

```
<btp:redirect id?>
  <btp:target-additional-information>
    ...additional address information...
  </btp:target-additional-information>
  <btp:superior-identifier>...hexstring...</btp:superior-
identifier> ?
  <btp:inferior-identifier>...hexstring...</btp:inferior-
identifier>
  <btp:old-address>  +
    ...address...
  </btp:old-address>
  <btp:new-address>  +
    ...address...
  </btp:new-address>
  <btp:qualifiers> ?
    ...qualifiers...
  </btp:qualifiers>
</btp:redirect>
```

FAULT

```
<btp:fault id?>
  <btp:target-additional-information>
    ...additional address information...
  </btp:target-additional-information>
  <btp:superior-identifier>...hexstring...</btp:superior-
identifier> ?
  <btp:inferior-identifier>...hexstring...</btp:inferior-
identifier> ?
  <btp:fault-type>...fault type name...</btp:fault-type>
  <btp:fault-data>...fault data...</btp:fault-data> ?
  <btp:qualifiers> ?
    ...qualifiers...
  </btp:qualifiers>
</btp:fault>
```

The following fault type names are represented by simple strings, corresponding to the entries
defined in the abstract message set:

- o   general
- o   unknown-parameter
- o   wrong-state
- o   communication-failure
- o   invalid-superior
- o   duplicate-inferior
- o   unknown-inferior

3362 Revisions of this specification may add other fault type names, which shall be simple strings
3363 of letters, numbers and hyphens. If other specifications define fault type name s to be used
3364 with BTP, the names shall be URIs.
3365
3366 Fault data can take on various forms:
3367
3368 Free text:
3369
3370
```
<btp:fault-data>...string data...</btp:fault-data>
```
3371
3372 Identifier:
3373
3374
```
<btp:fault-data>...hexstring...</btp:fault-data>
```
3375
3376
3377 Inferior Identity:
3378
3379
```
<btp:fault-data>
  <btp:inferior-address> +
    ...address...
  </btp:inferior-address>
  <btp:inferior-identifier>...hexstring...</btp:inferior-
identifier>
    </btp:fault-data>
```
3380
3381
3382
3383
3384
3385
3386
3387

Standard qualifiers
3389 The informal syntax for these messages assumes the namespace prefix "btpq" is associated
3390 with the URI "urn:oasis:names:tc:BTP:qualifiers".
3391

Transaction timelimit
3393
3394
```
<btpq:transaction-timelimit>
  <btpq:timelimit>
    ...time in seconds...
  </btpq:timelimit>
</btpq:transaction-timelimit>
```
3395
3396
3397
3398
3399

Inferior timeout
3401
```
        <btpq:inferior-timeout>
  <btpq:timeout>
    ...time in seconds...
  </btpq:timeout>
  <btpq:intended-decision>confirm|cancel</btpq:intended-decision>
</btpq:inferior-timeout>
```
3402
3403
3404
3405
3406
3407

Minimum inferior timeout
3409
```
        <btpq:minimum-inferior-timeout>
  <btpq:minimum-timeout>
    ...time in seconds...
  </btpq:minimum-timeout>
```
3410
3411
3412

```
3413            </btpq:minimum-inferior-timeout>
3414
```

Compounding of Messages

Bundling (semantically insignificant combination) of BTP messages is indicated with the
"btp:messages" element, with the bundled messages as child elements. For example:

```
        <btp:messages>
          <btp:enrol>...</btp:enrol>
          <btp:prepared>...</btp:prepared>
        </btp:messages>
```

Relating BTP messages to one another is achieved through containment. For example:

```
        <btp:context-reply>
            ...<completion-status>related</completion-status> ...
            <btp:enrol>...</btp:enrol>
        </btp:context-reply>
```

The carrier protocol binding specifies how a relation between application and BTP messages
is represented.

Carrier Protocol Bindings

3438

3439 The notion of bindings is introduced to act as the glue between the BTP XML messages and
3440 an underlying transport. A binding specification must define various particulars of how the
3441 BTP messages are carried and some aspects of how the related application messages are
3442 carried. This document specifies two bindings: a SOAP binding and a SOAP + Attachments
3443 binding. However, other bindings could be specified by the Oasis BTP technical committee
3444 or by a third party. For example, in the future a binding might exist to put a BTP message
3445 directly on top of HTTP without the use of SOAP, or a closed community could define their
3446 own binding. To ensure that such specifications are complete, the Binding Proforma defines
3447 the information that must be included in a binding specification.
3448

Carrier Protocol Binding Proforma

3450

3451 A BTP carrier binding specification should provide the following information:

3452

3453 **Binding name:** A name for the binding, as used in the "binding name" field of BTP
3454 addresses (and available for declaring the capabilities of an implementation). Binding
3455 specified in this document, and future revisions of this document have binding names that are
3456 simple strings of letters, numbers and hyphens (and, in particular, do not contain colons).
3457 Bindings specified elsewhere shall have binding names that are URIs. Bindings specified in
3458 this document use numbers to identify the version of the binding, not the version(s) of the
3459 carrier protocol.

3460

3461 **BTP message representation:** This section will define how BTP messages are
3462 rerpresented. For many bindings, this will be the normal string encoding of the XML,
3463 in accordance with the XML schema defined in this document.

3464

3465 **Mapping for BTP messages (unrelated)** : This section will define how BTP messages
3466 that are not related to application messages are sent in either direction between
3467 Superior and Inferior. (i.e. those messages sent directly between BTP actors). This
3468 mapping need not be symmetric (i.e. Superior to Inferior may differ to some degree to
3469 Inferior to Superior). The mapping may define particular rules for particular BTP
3470 messages, or messages with particular parameter values (e.g. the FAULT message
3471 with "fault-type" "CommunicationFailure" will typically not be sent as a BTP
3472 message). The mapping states any constraints or requirements on which BTP may or
3473 must be bundled together by compounding.

3474

3475 **Mapping for BTP messages related to application messages** : This section will define
3476 how BTP messages that are related to application messages are sent. A binding
3477 specification may defer details of this to a particular application (e.g. a mapping
3478 specification could just say "the CONTEXT may be carried as a parameter of an
3479 application invocation"). Alternatively, the binding may specify a general method that
3480 represents the relationship between application and BTP messages.

3481

Implicit messages: This section specifies which BTP messages, if any, are not sent explicitly but are treated as implicit in application messages or other BTP messages. This may depend on particular parameter values of the BTP messages or the application messages.

Faults: The relationship between the fault and exception reporting mechanisms of the carrier protocol and of BTP shall be defined. This may include definition of which carrier protocol exceptions are equivalent to a FAULT/communication-failure message.

Relationship to other bindings: Any relationship to other bindings is defined in this section. If BTP addresses with different bindings are be considered to match (for purposes of identifying the peer Superior/Inferior and redirection), this should be specified here.

Limitations on BTP use: Any limitations on the full range of BTP functionality that are imposed by use of this binding should be listed. This would include limitations on which messages can be sent, which event sequences are supported and restrictions on parameter values. Such limitations may reduce the usefulness of an implementation, but may be appropriate in certain environments.

Other: Other features of the binding, especially any that will potentially affect interoperation should be specified here. This may include restrictions or requirements on the use or support of optional carrier parameters or mechanisms>

SOAP Binding

This binding describes how BTP messages will be carried using SOAP as in the SOAP 1.1 specification.

Binding name: soap-http-1

BTP message representation: The string representation of the XML, as specified in the XML schema defined in this document shall be used. BTP messages conform to the rules of the Section 5 (of the SOAP 1.1 specification) SOAP Encoding as specified by the URI: "http://schemas.xmlsoap.org/soap/encoding/".

Mapping for BTP messages (unrelated): If no application message is being sent at the same time, BTP messages shall be contained in a btp:messages element which shall be an immediate child element of the SOAP-Body. There shall be precisely one btp:messages element. Any number of BTP messages with the same binding address in their target address may be carried in the same btp:messages element.

If an application message is being sent at the same time, the mapping for related messages shall be used, as if the BTP messages were related to the application message. (There is no ambiguity in whether the BTP messages are related, because only CONTEXT can be related to an application message.)

Mapping for BTP messages related to application messages: All BTP messages sent with an application message, whether related to the application message or not, shall be sent in a single btp:messages element in the SOAP:Header. There shall be precisely one btp:messages element in the SOAP:Header.

Implicit messages: A SOAP fault, or other communication failure received in response to a SOAP request that had a CONTEXT in the SOAP:Header shall be treated as if a CONTEXT_REPLY/repudiated had been received. See also the discussion under "other" about the SOAP mustUnderstand attribute.

Faults: A SOAP fault or other communication failure shall be treated as FAULT/communication-failure.

Relationship to other bindings: A BTP address for Superior or Inferior that has the binding string "soap-http-1" is considered to match one that has the binding string "soap-attachments-http-1" if the binding address and additional information fields match.

Limitations on BTP use: None

Other: The SOAP BTP binding does not make use of SOAPAction HTTP header or actor attribute. The SOAPAction HTTP header is left to be application specific when there are application messages in the SOAP:Body, as an already existing web service that is being upgraded to use BTP might have already made use of SOAPAction. The SOAPAction HTTP header shall be omitted when the SOAP message carries only BTP messages in the SOAP:Body.

The SOAP mustUnderstand attribute, when used on the btp:messages containing a the BTP CONTEXT, ensures that the server (as a whole) determines whether any enrolments are necessary and reply with CONTEXT_REPLY as appropriate. If mustUnderstand if false, a server can ignore the CONTEXT (if BTP is not supported there). It is an implementation or configuration option whether a CONTEXT_REPLY/ok is assumed to be implicit in the HTTP response in such a case. (If no CONTEXT_REPLY/ok is assumed, it will be impossible for the business transaction to confirm) .

> Note – some SOAP implementations may not support the mustUnderstand
> attribute sufficiently to enforce these requirements. If using such an
> implementation on the service side, it may be necessary to assume an
> CONTEXT_REPLY/ok.

Example scenario using SOAP binding

The example below shows an application request with CONTEXT message sent from client.example.com (which includes the Superior) to services.example.com (Service).

```
<soap:Envelope
```

```
3573              xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
3574              soap-
3575      env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
3576
3577        <soap:Header>
3578
3579          <btp:messages xmlns:btp="urn:oasis:names:tc:BTP:xml">
3580            <btp:context superior-type="atom">
3581              <btp:superior-address>
3582                <btp:binding>soap-http-1</btp:binding>
3583                <btp:binding-
3584      address>http://client.example.com/soaphandler</btp:binding-
3585      address>
3586                <btp:additional-information>btpengine</btp:additional-
3587      information>
3588              </btp:superior-address>
3589              <btp:superior-identifier>1001</btp:superior-identifier>
3590              <btp:qualifiers>
3591                <btpq:transaction-timelimit
3592      xmlns:btpq="urn:oasis:names:tc:BTP:qualifiers">1800</btpq:transact
3593      ion-timelimit>
3594              </btp:qualifiers>
3595            </btp:context>
3596          </btp:messages>
3597
3598        </soap:Header>
3599
3600        <soap:Body>
3601
3602          <ns1:orderGoods
3603      xmlns:ns1="http://example.com/2001/Services/xyzgoods">
3604            <custID>ABC8329045</custID>
3605            <itemID>224352</itemID>
3606            <quantity>5</quantity>
3607          </ns1:orderGoods>
3608
3609        </soap:Body>
3610
3611      </soap:Envelope>
3612
3613
```

3614   The example below shows CONTEXT_REPLY and a related (and therefore contained)
3615   ENROL message sent from services.example.com to client.example.com, in reply to the
3616   previous message. There is no application response, so the BTP messages are in the
3617   SOAP:Body.
3618

```
3619      <soap:Envelope
3620          xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
3621          soap-
3622      env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
3623
3624        <soap:Header>
3625        </soap:Header>
3626
```

```
3627                    <soap:Body>
3628
3629                      <btp:messages xmlns:btp="urn:oasis:names:tc:BTP:xml">
3630                         <btp:context-reply>
3631                           <btp:superior-address>
3632                             <btp:binding>soap-http-1</btp:binding>
3633                             <btp:binding-address>
3634                                 http://client.example.com/soaphandler
3635                             </btp:binding-address>
3636                             <btp:additional-information>
3637                                 btpengine
3638                             </btp:additional-information>
3639                           </btp:superior-address>
3640                           <btp:superior-identifier>1001</btp:superior-identifier>
3641                           <completion-status>related</completion-status>
3642
3643                           <btp:enrol reply-requested="false">
3644                             <btp:target-additional-information>
3645                                 btpengine
3646                             </btp:target-additional-information>
3647                             <btp:superior-identifier>
3648                                 1001
3649                             </btp:superior-identifier>
3650                             <btp:inferior-address>
3651                               <btp:binding>soap-http-1</btp:binding>
3652                               <btp:binding-address>
3653                                 http://services.example.com/soaphandler
3654                               </btp:binding-address>
3655                             </btp:inferior-address>
3656                             <btp:inferior-identifier>
3657                                 AAAB
3658                             </btp:inferior-identifier>
3659                           </btp:enrol>
3660
3661                         </btp:context-reply>
3662
3663                      </btp:messages>
3664
3665                    </soap:Body>
3666
3667                 </soap:Envelope>
3668
3669
3670
```

SOAP + Attachments Binding
3671

3672
3673 This binding describes how BTP messages will be carried using SOAP as in the SOAP
3674 Messages with Attachments specification. It is a superset of the Basic SOAP binding, soap-
3675 http-1. The two bindings only differ when application messages are sent
3676
3677 **Binding name**: soap-attachments-http-1
3678

BTP message representation: As for soap-http-1

Mapping for BTP messages (unrelated): As for "soap-http-1", except the SOAP:Envelope containing the SOAP-Body containing the BTP messages shall be in a MIME body part, as specified in <u>SOAP Messages with Attachments</u> specification. If an application message is being sent at the same time, the mapping for related messages for this binding shall be used, as if the BTP messages were related to the application message(s).

Mapping for BTP messages related to application messages: MIME packaging shall be used. One of the MIME multipart/related parts shall contain a SOAP:Envelope, whose SOAP:Headers element shall contain precisely one btp:messages element, containing any BTP messages. Any BTP CONTEXT in the btp:messages is considered to be related to the application message(s) in the SOAP:Body, and to also any of the MIME parts referenced from the SOAP:Body (using the "href" attribute).

Implicit messages: As for soap-http-1.

Faults: As for soap-http-1.

Relationship to other bindings: A BTP address for Superior or Inferior that has the binding string "soap-http-1" is considered to match one that has the binding string "soap-attachements-http-1" if the binding address and additional information fields match.

Limitations on BTP use: None

Other: As for soap-http-1

Example using SOAP + Attachments binding

```
MIME-Version: 1.0
Content-Type: Multipart/Related; boundary=MIME_boundary;
type=text/xml;
        start="someID"

--MIME_boundary
Content-Type: text/xml; charset=UTF-8
Content-ID: someID

<?xml version='1.0' ?>
<soap:Envelope
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    soap-
env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">

  <soap:Header>

    <btp:messages xmlns:btp="urn:oasis:names:tc:BTP:xml">
      <btp:context superior-type="atom">
```

```
3728              <btp:superior-address>
3729                <btp:binding>soap-http-1</btp:binding>
3730                <btp:binding-address>
3731                    http://client.example.com/soaphandler
3732                </btp:binding-address>
3733              </btp:superior-address>
3734            <btp:superior-identifier>1001</btp:superior-identifier>
3735          </btp:context>
3736        </btp:messages>
3737
3738      </soap:Header>
3739
3740      <soap:Body>
3741        <orderGoods href="cid:anotherID"/>
3742      </soap:Body>
3743
3744    </soap:Envelope>
3745
3746    --MIME_boundary
3747    Content-Type: text/xml
3748    Content-ID: anotherID
3749
3750        <ns1:orderGoods
3751    xmlns:ns1="http://example.com/2001/Services/xyzgoods">
3752        <custID>ABC8329045</custID>
3753        <itemID>224352</itemID>
3754        <quantity>5</quantity>
3755      </ns1:orderGoods>
3756
3757
3758    --MIME_boundary--
3759
3760
```

XML Schema for SOAP Bindings

```
3763  <?xml version="1.0"?>
3764  <schema targetNamespace="urn:oasis:names:tc:BTP:xml"
3765         xmlns="http://www.w3.org/2001/XMLSchema"
3766         xmlns:tns="urn:oasis:names:tc:BTP:xml">
3767
3768    <complexType name="qualifier_type">
3769        <simpleContent>
3770            <extension base="string">
3771                <attribute name="must-be-understood" type="boolean"/>
3772                <attribute name="to-be-propagated" type="boolean"/>
3773            </extension>
3774        </simpleContent>
3775    </complexType>
3776    <element name="qualifier" type="tns:qualifier_type"/>
3777    <element name="qualifiers">
3778        <complexType>
3779            <sequence>
3780                <element ref="tns:qualifier" maxOccurs="unbounded"/>
```

```
3781                    </sequence>
3782               </complexType>
3783          </element>
3784
3785          <complexType name="address">
3786               <sequence>
3787                    <element name="binding-name" type="string"/>
3788                    <element name="binding-address" type="string"/>
3789                    <element name="additional-information" type="string"
3790     minOccurs="0"/>
3791               </sequence>
3792          </complexType>
3793
3794          <simpleType name="identifier">
3795            <restriction base="string">
3796             <pattern value="([0-9,A-Z])*"/>
3797            </restriction>
3798          </simpleType>
3799
3800          <simpleType name="superior-type">
3801               <restriction base="string">
3802                    <enumeration value="cohesion"/>
3803                    <enumeration value="atom"/>
3804               </restriction>
3805          </simpleType>
3806
3807          <simpleType name="transaction-type">
3808               <restriction base="string">
3809                    <enumeration value="cohesion"/>
3810                    <enumeration value="atom"/>
3811               </restriction>
3812          </simpleType>
3813
3814
3815          <element name="context">
3816               <complexType>
3817                    <sequence>
3818                         <element name="superior-address" type="tns:address"
3819     maxOccurs="unbounded"/>
3820                         <element name="superior-identifier"
3821     type="tns:identifier"/>
3822                         <element ref="tns:qualifiers" minOccurs="0"/>
3823                    </sequence>
3824                    <attribute name="id" type="ID" use="optional"/>
3825                    <attribute name="superior-type" type="tns:superior-type"
3826     use="required"/>
3827               </complexType>
3828          </element>
3829
3830          <element name="context-reply">
3831               <complexType>
3832                    <sequence>
3833                         <element name="superior-address" type="tns:address"
3834     maxOccurs="unbounded"/>
```

```
                    <element name="superior-identifier"
type="tns:identifier"/>
                    <element name="completion-status">
                        <simpleType>
                            <restriction base="string">
                                <enumeration value="completed"/>
                                <enumeration value="related"/>
                                <enumeration value="repudiated"/>
                            </restriction>
                        </simpleType>
                    </element>
                    <element ref="tns:qualifiers" minOccurs="0"/>
                </sequence>
                <attribute name="id" type="ID"/>
                <attribute name="superior-type" type="tns:superior-type"
use="required"/>
            </complexType>
        </element>

        <element name="begin">
            <complexType>
                <sequence>
                    <element name="target-additional-information"
type="string"/>
                    <element name="reply-address" type="tns:address"/>
                    <element ref="tns:qualifiers" minOccurs="0"/>
                </sequence>
                <attribute name="id" type="ID"/>
                <attribute name="transaction-type" type="tns:superior-type"
use="required"/>
            </complexType>
        </element>

        <element name="begun">
            <complexType>
                <sequence>
                    <element name="target-additional-information"
type="string"/>
                    <element name="decider-address" type="tns:address"
minOccurs="0"/>
                    <element name="transaction-identifier"
type="tns:identifier" minOccurs="0"/>
                    <element name="inferior-handle" type="tns:identifier"
minOccurs="0"/>
                    <element name="inferior-address" type="tns:address"
minOccurs="0"/>
                    <element ref="tns:qualifiers" minOccurs="0"/>
                </sequence>
                <attribute name="id" type="ID"/>
                <attribute name="transaction-type" type="tns:superior-type"
use="required"/>
            </complexType>
        </element>
```

```
3889          <element name="enrol">
3890              <complexType>
3891                  <sequence>
3892                      <element name="target-additional-information"
3893     type="string"/>
3894                      <element name="superior-identifier"
3895     type="tns:identifier"/>
3896                      <element name="reply-address" type="tns:address"
3897     minOccurs="0"/>
3898                      <element name="inferior-address" type="tns:address"
3899     minOccurs="1" maxOccurs="unbounded"/>
3900                      <element name="inferior-identifier"
3901     type="tns:identifier"/>
3902                      <element ref="tns:qualifiers" minOccurs="0"/>
3903                  </sequence>
3904                  <attribute name="id" type="ID"/>
3905                  <attribute name="reply-requested" type="boolean"/>
3906              </complexType>
3907          </element>
3908
3909
3910          <element name="enrolled">
3911              <complexType>
3912                  <sequence>
3913                      <element name="target-additional-information"
3914     type="string"/>
3915                      <element name="inferior-identifier"
3916     type="tns:identifier"/>
3917                      <element name="inferior-handle" type="tns:identifier"
3918     minOccurs="0"/>
3919                      <element ref="tns:qualifiers" minOccurs="0"/>
3920                  </sequence>
3921                  <attribute name="id" type="ID"/>
3922              </complexType>
3923          </element>
3924
3925          <element name="resign">
3926              <complexType>
3927                  <sequence>
3928                      <element name="target-additional-information"
3929     type="string"/>
3930                      <element name="superior-identifier"
3931     type="tns:identifier"/>
3932                      <element name="inferior-address" type="tns:address"
3933     minOccurs="1" maxOccurs="unbounded"/>
3934                      <element name="inferior-identifier"
3935     type="tns:identifier"/>
3936                      <element ref="tns:qualifiers" minOccurs="0"/>
3937                  </sequence>
3938                  <attribute name="id" type="ID"/>
3939                  <attribute name="response-requested" type="boolean"/>
3940              </complexType>
3941          </element>
3942
```

```
3943        <element name="resigned">
3944            <complexType>
3945                <sequence>
3946                    <element name="target-additional-information"
3947    type="string"/>
3948                    <element name="inferior-identifier"
3949    type="tns:identifier"/>
3950                    <element ref="tns:qualifiers" minOccurs="0"/>
3951                </sequence>
3952                <attribute name="id" type="ID"/>
3953            </complexType>
3954        </element>
3955
3956        <element name="prepare">
3957            <complexType>
3958                <sequence>
3959                    <element name="target-additional-information"
3960    type="string"/>
3961                    <element name="inferior-identifier"
3962    type="tns:identifier" minOccurs="0"/>
3963                    <element name="reply-address" type="tns:address"
3964    minOccurs="0"/>
3965                    <element name="transaction-identifier"
3966    type="tns:identifier" minOccurs="0"/>
3967                    <element name="inferiors-list" minOccurs="0">
3968                        <complexType>
3969                            <sequence>
3970                                <element name="inferior-handle"
3971    type="tns:identifier" maxOccurs="unbounded"/>
3972                            </sequence>
3973                        </complexType>
3974                    </element>
3975                    <element ref="tns:qualifiers" minOccurs="0"/>
3976                </sequence>
3977                <attribute name="id" type="ID"/>
3978            </complexType>
3979        </element>
3980
3981        <element name="prepared">
3982            <complexType>
3983                <sequence>
3984                    <element name="target-additional-information"
3985    type="string"/>
3986                    <element name="superior-identifier"
3987    type="tns:identifier"/>
3988                    <element name="inferior-address" type="tns:address"
3989    maxOccurs="unbounded"/>
3990                    <element name="inferior-identifier"
3991    type="tns:identifier"/>
3992                    <element ref="tns:qualifiers" minOccurs="0"/>
3993                </sequence>
3994                <attribute name="id" type="ID"/>
3995                <attribute name="default-is-cancel" type="boolean"/>
3996            </complexType>
```

```
3997            </element>
3998
3999        <element name="confirm">
4000            <complexType>
4001                <sequence>
4002                    <element name="target-additional-information"
4003    type="string"/>
4004                    <element name="inferior-identifier"
4005    type="tns:identifier"/>
4006                    <element ref="tns:qualifiers" minOccurs="0"/>
4007                </sequence>
4008                <attribute name="id" type="ID"/>
4009            </complexType>
4010        </element>
4011
4012        <element name="confirmed">
4013            <complexType>
4014                <sequence>
4015                    <element name="target-additional-information"
4016    type="string"/>
4017                    <element name="superior-identifier"
4018    type="tns:identifier"/>
4019                    <element name="inferior-address" type="tns:address"
4020    minOccurs="0"/>
4021                    <element name="inferior-identifier"
4022    type="tns:identifier" minOccurs="0"/>
4023                    <element name="decider-address" type="tns:address"
4024    minOccurs="0"/>
4025                    <element name="transaction-identifier"
4026    type="tns:identifier" minOccurs="0"/>
4027                    <element ref="tns:qualifiers" minOccurs="0"/>
4028                </sequence>
4029                <attribute name="id" type="ID"/>
4030                <attribute name="confirmed-received" type="boolean"/>
4031            </complexType>
4032        </element>
4033
4034        <element name="cancel">
4035            <complexType>
4036                <sequence>
4037                    <element name="target-additional-information"
4038    type="string"/>
4039                    <element name="inferior-identifier"
4040    type="tns:identifier" minOccurs="0"/>
4041                    <element name="reply-address" type="tns:address"
4042    minOccurs="0"/>
4043                    <element name="transaction-identifier"
4044    type="tns:identifier" minOccurs="0"/>
4045                    <element name="decider-address" type="tns:address"
4046    minOccurs="0"/>
4047                    <element name="transaction-identifier"
4048    type="tns:identifier" minOccurs="0"/>
4049                    <element name="inferiors-list" minOccurs="0">
4050                        <complexType>
```

```
4051                            <sequence>
4052                                <element name="inferior-handle"
4053    type="tns:identifier" maxOccurs="unbounded"/>
4054                            </sequence>
4055                        </complexType>
4056                    </element>
4057                    <element ref="tns:qualifiers" minOccurs="0"/>
4058                </sequence>
4059                <attribute name="id" type="ID"/>
4060            </complexType>
4061        </element>
4062
4063        <element name="cancelled">
4064            <complexType>
4065                <sequence>
4066                    <element name="target-additional-information"
4067    type="string"/>
4068                    <element name="superior-identifier"
4069    type="tns:identifier"/>
4070                    <element name="inferior-address" type="tns:address"
4071    maxOccurs="unbounded"/>
4072                    <element name="inferior-identifier"
4073    type="tns:identifier" minOccurs="0"/>
4074                    <element name="decider-address" type="tns:address"
4075    minOccurs="0"/>
4076                    <element name="transaction-identifier"
4077    type="tns:identifier" minOccurs="0"/>
4078                    <element ref="tns:qualifiers" minOccurs="0"/>
4079                </sequence>
4080                <attribute name="id" type="ID"/>
4081            </complexType>
4082        </element>
4083
4084        <element name="hazard">
4085            <complexType>
4086                <sequence>
4087                    <element name="target-additional-information"
4088    type="string"/>
4089                    <element name="superior-identifier"
4090    type="tns:identifier"/>
4091                    <element name="inferior-address" type="tns:address"
4092    maxOccurs="unbounded"/>
4093                    <element name="inferior-identifier"
4094    type="tns:identifier"/>
4095                    <element ref="tns:qualifiers" minOccurs="0"/>
4096                </sequence>
4097                <attribute name="id" type="ID"/>
4098            </complexType>
4099        </element>
4100
4101        <element name="contradiction">
4102            <complexType>
4103                <sequence>
```

```
4104                    <element name="target-additional-information"
4105  type="string"/>
4106                    <element name="inferior-identifier"
4107  type="tns:identifier"/>
4108                    <element ref="tns:qualifiers" minOccurs="0"/>
4109                </sequence>
4110                <attribute name="id" type="ID"/>
4111            </complexType>
4112        </element>
4113
4114        <element name="superior-state">
4115            <complexType>
4116                <sequence>
4117                    <element name="target-additional-information"
4118  type="string"/>
4119                    <element name="inferior-identifier"
4120  type="tns:identifier"/>
4121                    <element name="status">
4122                        <simpleType>
4123                            <restriction base="string">
4124                                <enumeration value="active"/>
4125                                <enumeration value="prepared-received"/>
4126                                <enumeration value="inaccessible"/>
4127                                <enumeration value="unknown"/>
4128                            </restriction>
4129                        </simpleType>
4130                    </element>
4131                    <element ref="tns:qualifiers" minOccurs="0"/>
4132                </sequence>
4133                <attribute name="id" type="ID"/>
4134                <attribute name="reply-requested" type="boolean"/>
4135            </complexType>
4136        </element>
4137
4138        <element name="inferior-state">
4139            <complexType>
4140                <sequence>
4141                    <element name="target-additional-information"
4142  type="string"/>
4143                    <element name="superior-identifier"
4144  type="tns:identifier"/>
4145                    <element name="inferior-address" type="tns:address"
4146  maxOccurs="unbounded"/>
4147                    <element name="inferior-identifier"
4148  type="tns:identifier"/>
4149                    <element name="status">
4150                        <simpleType>
4151                            <restriction base="string">
4152                                <enumeration value="active"/>
4153                                <enumeration value="prepared-received"/>
4154                                <enumeration value="inaccessible"/>
4155                                <enumeration value="unknown"/>
4156                            </restriction>
4157                        </simpleType>
```

```
4158                    </element>
4159                    <element ref="tns:qualifiers" minOccurs="0"/>
4160               </sequence>
4161               <attribute name="id" type="ID"/>
4162               <attribute name="reply-requested" type="boolean"/>
4163          </complexType>
4164     </element>
4165
4166     <element name="confirm-one-phase">
4167          <complexType>
4168               <sequence>
4169                    <element name="target-additional-information"
4170  type="string"/>
4171                    <element name="inferior-identifier"
4172  type="tns:identifier"/>
4173                    <element ref="tns:qualifiers" minOccurs="0"/>
4174               </sequence>
4175               <attribute name="id" type="ID"/>
4176               <attribute name="report-hazard" type="boolean"/>
4177          </complexType>
4178     </element>
4179
4180     <element name="request-confirm">
4181          <complexType>
4182               <sequence>
4183                    <element name="target-additional-information"
4184  type="string"/>
4185                    <element name="reply-address" type="tns:address"/>
4186                    <element name="transaction-identifier"
4187  type="tns:identifier"/>
4188                    <element name="inferiors-list" minOccurs="0">
4189                         <complexType>
4190                              <sequence>
4191                                   <element name="inferior-handle"
4192  type="tns:identifier" maxOccurs="unbounded"/>
4193                              </sequence>
4194                         </complexType>
4195                    </element>
4196                    <element ref="tns:qualifiers" minOccurs="0"/>
4197               </sequence>
4198               <attribute name="id" type="ID"/>
4199               <attribute name="report-hazard" type="boolean"/>
4200          </complexType>
4201     </element>
4202
4203     <element name="request-statuses">
4204          <complexType>
4205               <sequence>
4206                    <element name="target-additional-information"
4207  type="string"/>
4208                    <element name="reply-address" type="tns:address"/>
4209                    <element name="transaction-identifier"
4210  type="tns:identifier"/>
4211                    <element name="inferiors-list" minOccurs="0">
```

```
                        <complexType>
                            <sequence>
                                <element name="inferior-handle"
type="tns:identifier" maxOccurs="unbounded"/>
                            </sequence>
                        </complexType>
                    </element>
                    <element ref="tns:qualifiers" minOccurs="0"/>
            </sequence>
            <attribute name="id" type="ID"/>
        </complexType>
    </element>

    <element name="inferior-statuses">
        <complexType>
            <sequence>
                <element name="target-additional-information"
type="string"/>
                <element name="decider-address" type="tns:address"/>
                <element name="transaction-identifier"
type="tns:identifier"/>
                <element name="status-list">
                  <complexType>
                    <sequence>
                      <element name="status-item" maxOccurs="unbounded">
                      <complexType>
                        <sequence>
                          <element name="inferior-handle"
type="tns:identifier"/>
                            <element name="status">
                              <simpleType>
                              <restriction base="string">
                                  <enumeration value="active"/>
                                  <enumeration value="resigned"/>
                                  <enumeration value="preparing"/>
                                  <enumeration value="prepared"/>
                                  <enumeration value="autonomously-
confirmed"/>
                                  <enumeration value="autonomously-
cancelled"/>
                                  <enumeration value="confirming"/>
                                  <enumeration value="confirmed"/>
                                  <enumeration value="cancelling"/>
                                  <enumeration value="cancelled"/>
                                  <enumeration value="cancel-contradiction"/>
                                  <enumeration value="confirm-contradiction"/>
                                  <enumeration value="hazard"/>
                              </restriction>
                                </simpleType>
                              </element>
                              <element ref="tns:qualifiers" minOccurs="0"/>
                          </sequence>
                        </complexType>
                          </element>
```

```
4266                         </sequence>
4267                       </complexType>
4268                     </element>
4269                     <element ref="tns:qualifiers" minOccurs="0"/>
4270               </sequence>
4271               <attribute name="id" type="ID"/>
4272          </complexType>
4273     </element>
4274
4275     <element name="request-status">
4276          <complexType>
4277               <sequence>
4278                    <element name="target-additional-information"
4279     type="string"/>
4280                    <element name="reply-address" type="tns:address"/>
4281                    <element name="inferior-identifier"
4282     type="tns:identifier" minOccurs="0"/>
4283                    <element name="transaction-identifier"
4284     type="tns:identifier" minOccurs="0"/>
4285                    <element ref="tns:qualifiers" minOccurs="0"/>
4286               </sequence>
4287               <attribute name="id" type="ID"/>
4288          </complexType>
4289     </element>
4290
4291     <element name="status">
4292          <complexType>
4293               <sequence>
4294                    <element name="target-additional-information"
4295     type="string"/>
4296                    <element name="inferior-address" type="tns:address"
4297     minOccurs="0"/>
4298                    <element name="inferior-identifier"
4299     type="tns:identifier" minOccurs="0"/>
4300                    <element name="decider-address" type="tns:address"
4301     minOccurs="0"/>
4302                    <element name="transaction-identifier"
4303     type="tns:identifier" minOccurs="0"/>
4304                    <element name="status-value">
4305                         <simpleType>
4306                         <restriction base="string">
4307                              <enumeration value="created"/>
4308                              <enumeration value="enrolling"/>
4309                              <enumeration value="active"/>
4310                              <enumeration value="resigning"/>
4311                              <enumeration value="resigned"/>
4312                              <enumeration value="preparing"/>
4313                              <enumeration value="prepared"/>
4314                              <enumeration value="confirming"/>
4315                              <enumeration value="confirmed"/>
4316                              <enumeration value="cancelling"/>
4317                              <enumeration value="cancelled"/>
4318                              <enumeration value="cancel-contradiction"/>
4319                              <enumeration value="confirm-contradiction"/>
```

```
4320                              <enumeration value="hazard"/>
4321                              <enumeration value="contradicted"/>
4322                              <enumeration value="unknown"/>
4323                              <enumeration value="inaccessible"/>
4324                          </restriction>
4325                        </simpleType>
4326                  </element>
4327                  <element ref="tns:qualifiers" minOccurs="0"/>
4328              </sequence>
4329              <attribute name="id" type="ID"/>
4330          </complexType>
4331      </element>
4332
4333      <element name="redirect">
4334          <complexType>
4335              <sequence>
4336                  <element name="target-additional-information"
4337  type="string"/>
4338                  <element name="superior-identifier"
4339  type="tns:identifier" minOccurs="0"/>
4340                  <element name="inferior-identifier"
4341  type="tns:identifier"/>
4342                  <element name="old-address" type="tns:address"
4343  maxOccurs="unbounded"/>
4344                  <element name="new-address" type="tns:address"
4345  maxOccurs="unbounded"/>
4346                  <element ref="tns:qualifiers" minOccurs="0"/>
4347              </sequence>
4348              <attribute name="id" type="ID"/>
4349          </complexType>
4350      </element>
4351
4352      <element name="fault">
4353          <complexType>
4354              <sequence>
4355                  <element name="target-additional-information"
4356  type="string"/>
4357                  <element name="superior-identifier"
4358  type="tns:identifier" minOccurs="0"/>
4359                  <element name="inferior-identifier"
4360  type="tns:identifier" minOccurs="0"/>
4361                  <element name="fault-type" type="string"/>
4362                  <element name="fault-data" type="anyType"
4363  minOccurs="0"/>
4364                  <element ref="tns:qualifiers" minOccurs="0"/>
4365              </sequence>
4366              <attribute name="id" type="ID"/>
4367          </complexType>
4368      </element>
4369
4370  </schema>
4371
```

4371
4372

Conformance

4373

4374
4375 A BTP implementation need not implement all aspects of the protocol to be useful. The level
4376 of conformance of an implementation is defined by which roles it can support using the
4377 specified messages and carrier protocol bindings for interoperation with other
4378 implementations.
4379
4380 A partially conformant implementation may implement some roles in a non-interoperable
4381 way, giving that implementation's users comparable proprietary functionality.
4382
4383 The following Roles and Role Groups are used to define conformance:
4384

Role Group	Role
Initiator/Terminator	Initiator
	Terminator
Cohesive Hub	Factory
	Composer (as Decider and Superior)
	Coordinator (as Decider and Superior)
	Sub-composer
	Sub-coordinator
Atomic Hub	Factory
	Coordinator
	Sub-coordinator
Cohesive Superior	Composer (as Superior only)
	Sub-Composer
	Coordinator (as Superior only)
	Sub-coordinator
Atomic Superior	Coordinator (as Superior only))
	Sub-coordinator
Participant	Inferior
	Enroller

4385

4386 An implementation may support one or more Role Groups. The following combinations are
4387 defined as commonly expected conformance profiles, although other combinations or
4388 selections are equally possible.
4389

Conformance Profile	Role Groups
Participant Only	Participant
Atomic	Atomic Superior Participant
Cohesive	Full Superior Participant
Atomic Coordination Hub	Initiator/Terminator Atomic Coordination Hub Participant
Cohesive Coordination Hub	Initiator/Terminator Cohesive Coordination Hub Participant

4390
4391
4392 BTP has several features, such as optional parameters, that allow alternative implementation
4393 architectures. Implementations should pay particular attention to avoid assuming their peers
4394 have made the same implementation options as they have (e.g. an implementation that always
4395 sends ENROL with the same inferior address and with the reply address absent (because the
4396 Inferior in all transactions are dealt with by the same addressable entity), must not assume
4397 that the same is true of received ENROLs)
4398

4399

Part 3. Appendices

> *These terms seem to be all either not used, or effectively defined elsewhere*

A. Glossary

Message	A datum which is produced and then consumed.
Sender	The producer of a message.
Receiver	The consumer of a message.
Transmission	The passage of a message from a sender to a receiver.
Endpoint	A sender or receiver.
Address	An identifier for an endpoint.
Carrier Protocol	A protocol which defines how transmissions occur.
Carrier Protocol Address **(CPA)**	The address of an endpoint for a particular carrier protocol.
Business Transaction Protocol Address **(BTPA)**	A compound address consisting of a mandatory *carrier protocol address* and an optional opaque suffix. > *PRF - suffix ? I've used "additional information"*
Actor	An entity which executes procedures, a software agent.
Application	An actor which uses the Business Transaction Protocol.
Application Message	A message produced by an application and consumed by an application.
Application Endpoint	An endpoint of an application message.
Operation	A procedure which is started by a receiver when a message arrives at it.

message arrives at it.

Application Operation	An operation which is started when an application message arrives.
Contract	Any rule, agreement or promise which constrains an actor's behaviour and is known to any other actor, and upon which any other knowing actor may rely.
Appropriate	In accordance with a pertinent contract.
Inappropriate	In violation of a pertinent contract.
Service	An actor which on receipt of an application messages may start an application operation which is appropriate. For example, a process which advertises an interface allowing defined RPCs to be invoked by a remote client.
Client	An actor which sends application messages to services.
Effect	The changes induced by the incomplete or complete processing of a set of procedures by an actor, which are observable by another contemporary or future actor, and which are made in conformance with a contract known to any such observer. This contract must state the countereffect of the effect, and is known as the countereffect contract. An effect is **Completed** when the change-inducing processing of the set of procedures is finished. [Need an indirect or consequential damage exclusion clause]

> *PRF - Sentence about countereffect contract doesn't fit well*

Ineffectual	Describes a set of procedures which has no effect.
Countereffect	An appropriate effect intended to counteract a prior effect.
Countereffect Contract	The contract which governs the relationship between the effect and the countereffect of a procedure. In the absence of any other overriding contracts the countereffect contract is the promise that
	"The **Countereffect** will attempt so far as is

possible to reverse or cancel the **Effect** such that an observer (on completion of the **Countereffect**) is unaware that the **Effect** ever occurred, but this attempt cannot be guaranteed to succeed".

Cancel	Process a countereffect for the current effect of a set of procedures.
Confirm	Ensure that the effect of a set of procedures is completed.
Prepare	Ensure that of a set of procedures is capable of being successfully instructed to cancel or to confirm.
Outcome	A decision to either cancel or confirm.
Participant	A set of procedures which is capable of receiving instructions from a coordinator to prepare, cancel and confirm. A participant must also have a BTPA to which these instructions will be delivered, in the form of BTP messages. A participant is identified by a participant identifier.
Inferior Identifier	An identifier assigned to an Inferior which is unique within the scope of an Address-as-Inferior.
Atomic Business Transaction *or* **Atom**	A set of participants (which may have only one member), all of which will receive instructions that will result in a homogeneous outcome. (Transitively, a set of operations, whose effect is capable of countereffect.) An atom is identified by an atom identifier.
Atom Identifier	A globally unique identifier assigned to an atom.

> *PRF – abs msgs define as unambiguous in scope of its address-as-superior, I think.*

Coordinator	An actor which decides the outcome of a single atom, and has a lifetime which is coincident with that of the atom. A coordinator can issue instructions to a participant to prepare, cancel and confirm. These instructions take the form of BTP messages. A coordinator is identified by its atom's atom identifier. A coordinator must also have a BTPA to which participants can send BTP messages.

Address-as-Superior	The address used to communicate with an actor playing the role of an Superior
Address-as-Composer	The address used to communicate with a Composer by an application actor that controls its resolution. The messages that might be sent to or received from this endpoint are undefined.
Address-as-Inferior	The address used to communicate with an actor playing the role of an Inferior.
Identity-as-Superior	The combination of Superior Identifier and Address-as-Superior of a given Superior.
Identity-as-Inferior	The combination of Inferior Identifier and Address-as-Inferior of a given Inferior.

4405