

1 Organization for the Advancement of Structured Information Systems

2 Business Transaction Protocol

3
4
5 An OASIS Committee Specification

6 ***CURRENT STATUS : internal committee draft***

7
8 Version 1.0 [0.9.0.4]

9 DD Mmm 2001 [12 January 2002 18:03]

10
11

<i>Working draft 0.1 (pre-London)</i>	14 June 2001
<i>Working draft 0.2 (London)</i>	18 June 2001
<i>Working draft 0.3a (circulated)</i>	12 July 2001
<i>Working draft 0.3c (circulated)</i>	20 July 2001
<i>Working draft 0.4 (circulated; incorporates PRF material)</i>	25 July 2001
<i>Working draft 0.6 (State tables)</i>	31 August 2001
<i>Working Draft 0.9</i>	24 October 2001
<i>Working Draft 0.9.0.1 – minor editorials issues applied</i>	16 November 2001
<i>Working Draft 0.9.0.2 – issue resolutions balloting to 10 Dec 2001</i>	4 December 2001
<i>Working Draft 0.9.0.3 – possible solution to msging issues</i>	11 December 2001
<i>Working Draft 0.9.0.4 – issue 79 solution, revise msging issues</i>	12 January 2002

12
13 [*Change marks relative to 0.9.0.2*](#)

14 Copyright and related notices

15
16 Copyright © The Organization for the Advancement of Structured Information Standards
17 (OASIS), 2001. All Rights Reserved.

18
19 This document and translations of it may be copied and furnished to others, and derivative
20 works that comment on or otherwise explain it or assist in its implementation may be
21 prepared, copied, published and distributed, in whole or in part, without restriction of any
22 kind, provided that the above copyright notice and this paragraph are included on all such
23 copies and derivative works. However, this document itself may not be modified in any way,
24 such as by removing the copyright notice or references to OASIS, except as needed for the
25 purpose of developing OASIS specifications, in which case the procedures for copyrights
26 defined in the OASIS Intellectual Property Rights document must be followed, or as required
27 to translate it into languages other than English.

28
29 The limited permissions granted above are perpetual and will not be revoked by OASIS or its
30 successors or assigns.

31
32 This document and the information contained herein is provided on an "AS IS" basis and
33 OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT
34 NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION
35 HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF
36 MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

37
38
39 OASIS takes no position regarding the validity or scope of any intellectual property or other
40 rights that might be claimed to pertain to the implementation or use of the technology
41 described in this document or the extent to which any license under such rights might or
42 might not be available; neither does it represent that it has made any effort to identify any
43 such rights. Information on OASIS's procedures with respect to rights in OASIS
44 specifications can be found at the OASIS website. Copies of claims of rights made available
45 for publication and any assurances of licenses to be made available, or the result of an attempt
46 made to obtain a general license or permission for the use of such proprietary rights by
47 implementors or users of this specification, can be obtained from the OASIS Executive
48 Director.

49
50 OASIS invites any interested party to bring to its attention any copyrights, patents or patent
51 applications, or other proprietary rights which may cover technology that may be required to
52 implement this specification. Please address the information to the OASIS Executive
53 Director.

54

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.

99 **Typographical and Linguistic Conventions and Style**

100
101 The initial letters of words in terms which are defined (at least in their substantive or
102 infinitive form) in the Glossary are capitalized whenever the term used with that exact
103 meaning, thus:

104
105 Cancel
106 Participant
107 Application Message
108

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

110 **Coordinator**

111
112 Such words may be given in bold in other contexts (for example, in section headings or
113 captions) to emphasize their status as formally defined terms.

114
115 The names of abstract BTP protocol messages are given in upper-case throughout:

116
117
118 BEGIN
119 CONTEXT
120 RESIGN
121

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

123
124 BEGIN/atom
125

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

127
128 BEGIN/atom & CONTEXT
129

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

131
132 ENROL + VOTE
133

134 XML schemata and instances are given in Courier:

135
136 <ctp:begin> ... </ctp:begin>
137

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

139
140 **int main (String[] args)**
141 **{**
142 **}**
143

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

147
148
149
150
151

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

151	Contents	
152		
153	Copyright and related notices	2
154	Acknowledgements	3
155	Typographical and Linguistic Conventions and Style	4
156	Contents.....	6
157	Part 1. Purpose and Features of BTP.....	10
158	Introduction	10
159	Development and Maintenance of the Specification	11
160	Overview of the Business Transaction Protocol.....	12
161	Part 2. Normative Specification of BTP	15
162	Actors, Roles and Relationships.....	15
163	Relationships	15
164	Roles involved in the outcome relationships	17
165	Superior	17
166	Inferior	18
167	Enroller.....	19
168	Participant	20
169	Sub-coordinator.....	20
170	Sub-composer	21
171	Roles involved in the control relationships	21
172	Decider.....	21
173	Coordinator	22
174	Composer.....	22
175	Terminator	23
176	Initiator	23
177	Factory	24
178	Other roles	24
179	Redirector.....	24
180	Status Requestor.....	25
181	Abstract Messages and Associated Contracts.....	25
182	Addresses.....	26
183	Request/response pairs.....	27
184	Compounding messages.....	28
185	Extensibility	30
186	Inferior handle	30
187	Messages	30
188	Qualifiers.....	30
189	Messages not restricted to outcome or control relationships.....	31
190	CONTEXT.....	32
191	CONTEXT_REPLY	33
192	REQUEST_STATUS	34
193	STATUS.....	35
194	FAULT.....	37
195	REQUEST_INFERIOR_STATUSES, INFERIOR_STATUSES.....	40
196	Messages used in the outcome relationships	40

197	ENROL	40
198	ENROLLED	41
199	RESIGN	42
200	RESIGNED	43
201	PREPARE.....	44
202	PREPARED.....	45
203	CONFIRM.....	47
204	CONFIRMED	47
205	CANCEL.....	49
206	CANCELLED.....	51
207	CONFIRM_ONE_PHASE.....	52
208	HAZARD	53
209	CONTRADICTION.....	54
210	SUPERIOR_STATE.....	55
211	INFERIOR_STATE.....	57
212	REDIRECT.....	61
213	Messages used in control relationships	62
214	BEGIN	62
215	BEGUN.....	63
216	PREPARE_INFERIORS.....	64
217	CONFIRM_TRANSACTION.....	65
218	TRANSACTION_CONFIRMED	67
219	CANCEL_TRANSACTION.....	68
220	CANCEL_INFERIORS.....	69
221	TRANSACTION_CANCELLED.....	71
222	REQUEST_INFERIOR_STATUSES.....	72
223	INFERIOR_STATUSES.....	73
224	Groups – combinations of related messages.....	78
225	CONTEXT & application message	78
226	CONTEXT_REPLY & ENROL.....	79
227	CONTEXT_REPLY (& ENROL) & PREPARED / & CANCELLED.....	80
228	CONTEXT_REPLY & ENROL & application message (& PREPARED).....	80
229	BEGUN & CONTEXT	81
230	BEGIN & CONTEXT	81
231	Standard qualifiers.....	81
232	Transaction timelimit	81
233	Inferior timeout	82
234	Minimum inferior timeout.....	83
235	Inferior name.....	84
236	State Tables.....	85
237	Explanation of the state tables	85
238	Status queries	85
239	Decision events	85
240	Disruptions – failure events.....	86
241	Invalid cells and assumptions of the communication mechanism.....	86
242	Meaning of state table events.....	87
243	Persistent information	91

244	Failure Recovery.....	104
245	Types of failure.....	104
246	Persistent information.....	105
247	Redirection.....	106
248	Terminator:Decider failures.....	107
249	XML representation of Message Set.....	107
250	Addresses.....	108
251	Qualifiers.....	108
252	Identifiers.....	109
253	Message References.....	109
254	Messages.....	109
255	CONTEXT.....	109
256	CONTEXT_REPLY.....	109
257	BEGIN.....	110
258	BEGUN.....	110
259	ENROL.....	110
260	ENROLLED.....	111
261	RESIGN.....	111
262	RESIGNED.....	112
263	PREPARE.....	112
264	PREPARED.....	112
265	CONFIRM.....	113
266	CONFIRMED.....	113
267	CANCEL.....	113
268	CANCELLED.....	114
269	CONFIRM_ONE_PHASE.....	114
270	HAZARD.....	115
271	CONTRADICTION.....	115
272	SUPERIOR_STATE.....	115
273	INFERIOR_STATE.....	115
274	REDIRECT.....	117
275	PREPARE_INFERIORS.....	117
276	CONFIRM_TRANSACTION.....	118
277	TRANSACTION_CONFIRMED.....	118
278	CANCEL_TRANSACTION.....	119
279	CANCEL_INFERIORS.....	119
280	TRANSACTION_CANCELLED.....	119
281	REQUEST_INFERIOR_STATUSES.....	120
282	INFERIOR_STATUSES.....	120
283	REQUEST_STATUS.....	121
284	STATUS.....	121
285	FAULT.....	122
286	Standard qualifiers.....	123
287	Transaction timelimit.....	123
288	Inferior timeout.....	123
289	Minimum inferior timeout.....	123
290	Compounding of Messages.....	124

291	Carrier Protocol Bindings	125
292	Carrier Protocol Binding Proforma	125
293	Bindings for request/response carrier protocols.....	126
294	Request/response exploitation rules	127
295	SOAP Binding.....	128
296	Example scenario using SOAP binding	130
297	SOAP + Attachments Binding.....	132
298	XML Schema	134
299	Conformance.....	146
300	Part 3. Appendices	148
301	A. Glossary	148
302		
303		

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.

346 **Development and Maintenance of the Specification**

347
348 For more information on the genesis and development of BTP, please consult the OASIS BT
349 Technical Committee's website, at

350
351 <http://www.oasis-open.org/committees/business-transactions/>
352

353
354 As of the date of adoption of this specification the OASIS BT Technical Committee is still in
355 existence, with the charter of

- 356
- 357 maintaining the specification in the light of implementation experiences
 - 358 coordinating publicity for BTP
 - 359 liaising with other standards bodies whose work affects or may be affected by
 - 360 BTP
 - 361 liaising with other standards bodies whose work affects or may be affected by
 - 362 BTP
 - 363 reviewing the appropriate time, in the light of implementation experience and
 - 364 user support, to put BTP forward for adoption as a full OASIS standard
 - 365
 - 366

367
368 If you have a question about the functionality of BTP, or wish to report an error or to suggest
369 a modification to the specification, please subscribe to:

370
371 bt-spec@lists.oasis-open.org
372

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

375
376 The main list of the committee is:

377
378 business-transaction@lists.oasis-open.org
379
380
381
382
383
384

384 Overview of the Business Transaction Protocol

385
386 A Business Transaction is a consistent change in the state of a business relationship between
387 two or more parties. BTP provides means to allow the consistent and coordinated changes in
388 the relationship as viewed from each party.

389
390 BTP assumes that for a given business transaction state changes occur, or are desired, in some
391 set of parties, and that these changes are related in some business-defined manner.

392
393 Typically business-defined messages (“application messages”) are exchanged between the
394 parties to the transaction, which result in the performance of some set of operations. These
395 operations create provisional or tentative state changes (the transaction’s effect). The
396 provisional changes of each party must either be confirmed (given final effect), or must be
397 cancelled (counter-effected). Those parties which are confirmed create an atomic unit, within
398 which the business transaction should have a consistent final effect.

399
400 The meaning of “effect”, “final effect” and “counter-effect” is specific to each business
401 transaction and to each party’s role within it. A party may log intended changes (as its effect)
402 and only process them as visible state changes on confirmation (its final effect). Or it may
403 make visible state changes and store the information needed to cancel (its effect), and then
404 simply delete the information needed for cancellation (its final effect). A counter-effect may
405 be a precise inversion or removal of provisional changes, or it may be the processing of
406 operations that in some way compensate for, make good, alleviate or supplement their effect.

407
408 To ensure that confirmation or cancellation of the provisiona l effect within different parties
409 can be consistently performed, it is necessary that each party should

- 410
411 determine whether it is able both to cancel (counter-effect) and to confirm (give final
412 effect to) its effect
- 413
414 report its ability or inability to cancel-or-confirm (its preparedness) to a central
415 coordinating entity

416
417 After receiving these reports, the coordinating entity is responsible for determining which of
418 the parties should be instructed to confirm and which should be instructed to cancel.

419
420 Such a two-phase exchange (ask, instruct) mediated by a central coordinator is required to
421 achieve a consistent outcome for a set of operations. BTP defines the means for software
422 agents executing on network nodes to interoperate using a two-phase coordination protocol,
423 leading either to the abandonment of the entire attempted transaction, or to the selection of an
424 internally consistent set of confirmed operations.

425
426 BTP centres on the bilateral relationship between the computer systems of the coordinating
427 entity and those of one of the parties in the overall business transaction. In that relationship a
428 software agent within the coordinating entity’s systems plays the BTP role of Superior for a
429 given transaction and one or more software agents within the systems of the party play the
430 BTP role of Inferior. Each Inferior has one Superior, therefore, while a single Superior may

431 have multiple Inferiors within each party to the transaction, and may be related to Inferiors
432 within multiple parties. Each Superior:Inferior pair exchanges protocol-defined messages.

433
434 An Inferior is associated with some set of operation invocations that creates effect
435 (provisional or tentative changes) within the party, for a given business transaction. The
436 Inferior is responsible for reporting to its related Superior whether its associated operations'
437 effect can be confirmed/cancelled. A Superior is responsible for gathering the reports of all of
438 its Inferiors, in order to ascertain which should be cancelled or confirmed. For example, if a
439 Superior is acting as an atomic Coordinator it will treat any Inferior which cannot prepare to
440 cancel/confirm as having veto power over the whole business transaction, causing the
441 Superior to instruct all its Inferiors to cancel. A Superior may, under the dictates of a
442 controlling application, increase or reduce the set of Inferiors to which a common confirm or
443 cancel outcome may be delivered. Thus, the set of prepared Inferiors may be larger than the
444 set of confirmed Inferiors.

445
446 An Inferior:Superior relationship is typically established in relation to one or more
447 application messages sent from one part of the application (linked to the Superior) to some
448 other part of the application to request the performance of operations that are to be subject to
449 the confirm or cancel decision of the Superior. If an application is divided between a client
450 and a service, which use RPCs to communicate application requests and responses, then the
451 client would typically be associated with the Superior and the service would typically host the
452 Inferior(s). (BTP does not mandate such an application topology nor does it require the use of
453 RPC or any other application communication paradigm.)

454
455 BTP defines a CONTEXT message that can be sent "in relation to" such application
456 messages. On receipt of a CONTEXT, one or more Inferiors may be created and "enrolled"
457 with the Superior, establishing the Superior:Inferior relationships. The particular mechanisms
458 by which a CONTEXT is "related" to application messages is an issue for the application
459 protocol and its binding to carrier mechanisms. BTP does not require that the enrolment is
460 requested by any particular entity – in a particular implementation this may be done by the
461 Inferior itself, by parts of the application or by other entities involved in the transmission of
462 the CONTEXT and the application messages. BTP defines a CONTEXT_REPLY message
463 that can be sent on the return path of the CONTEXT to indicate whether the enrolment was
464 successful. Without CONTEXT_REPLY it would be possible for a Superior to have an
465 incorrect view of which Inferiors it was supposed to involve in its confirm decision.

466
467 It should be noted that this BTP specification recognises that:

- 468 ❑ an Inferior may itself be a Superior to other BTP Inferiors; this occurs when some of
469 the operations associated with the Inferior involve other application elements whose
470 operations are to be subject to the confirm/cancel instruction sent to the Inferior. The
471 specification treats any lower Inferiors as part of the associated operations;
- 472 ❑ the requirement on an Inferior to be able to confirm or cancel does not include any
473 specific mechanism to determine the isolation of the effects of operations; the
474 requirement is only that the Inferior is able to confirm or cancel the operations, as
475 their effects are known to the Superior and the application directly in contact with the
476 Superior. Thus the confirm-or-cancel requirement may be achieved by performing all
477 the operations and remembering a compensating counter operation (that will be

478 triggered by a cancel order); or by remembering the operations (having checked they
479 are valid) and performing them only if a confirm order is received; or by forbidding
480 any other access to data changed by the operations and releasing them in their
481 unchanged state (if cancelled) or their changed state (if confirmed); or by various
482 combinations of these. In addition, a cancellation may not return data to their original
483 state, but only to a state accepted by the application as appropriate to a cancelled
484 operation.
485
486
487
488
489
490
491

491 Part 2. Normative Specification of BTP

492

493 **Actors, Roles and Relationships**

494

495 Actors are software agents which process computations. BTP actors are addressable for the
496 purposes of receiving application and BTP protocol messages transmitted over some
497 underlying communications or carrier protocol. (See section “Addressing” for more detail.)
498

499 BTP actors play roles in the sending, receiving and processing of messages. These roles are
500 associated with responsibilities or obligations under the terms of software contracts defined
501 by this specification. (These contracts are stated formally in the sections entitled “Abstract
502 Messages and Associated Contracts” and “State Tables”.) A BTP actor’s computations put
503 the contracts into effect.
504

505 A role is defined and described in terms of a single business transaction. An implementation
506 supporting a role may, as an addressable entity, play the same role in multiple business
507 transactions, simultaneously or consecutively, or a separate addressable entity may be created
508 for each transaction. This is a choice for the implementer, and the addressing mechanisms
509 allow interoperation between implementations that make different choices.
510

511 Within a single transaction, one actor may play several roles, or each role may be assigned to
512 a distinct actor. This is again a choice for the implementer. An actor playing a role is termed
513 an “actor-in-role”.
514

515 Actors may interoperate, in the sense that the roles played by actors may be implemented
516 using software created by different vendors for each actor-in-role. The section
517 “Conformance”, gives guidelines on the groups of roles that may be implemented in a
518 partial, interoperable implementation of BTP.
519

520 The descriptions of the roles concentrate on the normal progression of a business transaction,
521 and some of the more important divergences from this. They do not cover all exception cases
522 – the message set definition and the state tables provide a more comprehensive specification.
523

524

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.

525

526

527

528 **Relationships**

529

530 There are two primary relationships in BTP.

531

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

534

535 □ Between BTP actors within the tree, where one (the Superior) will inform the other
536 (the Inferior) what the outcome decision is.

537

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

- 541 1. The Terminator determines that the business transaction should confirm, if it can; or
542 (for a Cohesion), which parts should confirm
- 543 2. The Terminator asks the Decider to apply the desired outcome to the tree, if it can
544 guarantee the consistency of the confirm decision
- 545 3. The Decider, which is Superior to one or more Inferiors, asks its Inferiors if they can
546 agree to a confirm decision (for a Cohesion, this may not be all the Inferiors)
- 547 4. If any of those Inferiors are also Superiors, they ask their Inferiors and so on down
548 the tree
- 549 5. Inferiors that are not Superiors report if they can agree to a confirm to their Superior
- 550 6. Inferiors that are also Superiors report their agreement only if they received such
551 agreement from their Inferiors, and can agree themselves
- 552 7. Eventually agreement (or not) is reported to the Decider. If all have agreed, the
553 Decider makes and persists the confirm decision (hence the term “Decider” – it
554 decides, everything else just asked); if any have disagreed, or if the confirm decision
555 cannot be persisted, a cancel decision is made
- 556 8. The Decider, as Superior tells its Inferiors of the outcome
- 557 9. Inferiors that are also Superiors tell their Inferiors, recursively down the tree
- 558 10. The Decider replies to the Terminator’s request to confirm, reporting the outcome
559 decision

560

561 There are other relationships that are secondary to Terminator:Decider, Superior:Inferior,
562 mostly involved in the establishment of the primary relationships. The various particular
563 relationships can be grouped as the “control” relationships – primarily Terminator:Decider,
564 but also Initiator:Factory; and the “outcome” relationships – primarily Superior:Inferior, but
565 also Enroller:Superior.

566

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

570

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

574

- 575 2. The Superior:Inferior relationship is recoverable – depending on the progress of the
576 relationship, the two sides will re-establish their shared state after failure; the
577 Terminator:Decider relationship is not recoverable
578
- 579 3. The nature of the Superior:Inferior relationship requires that the two parties know of
580 each other’s addresses from when the relationship is established; the Decider does not
581 need to know the address of the Terminator (provided it has some way of returning
582 the response to a received message).
583

584 In the following sections, the responsibility of each role is defined, and the messages that are
585 sent or received by that role are listed. Note that some roles exist only to have a name for an
586 actor that issues a message and receives a reply to that message. Some of these roles may be
587 played by several actors in the course of a single business transaction.
588

589 Roles involved in the **Superior:Inferioroutcome** relationships

590 Superior

591
592 Accepts enrolments from Inferiors, establishing a Superior:Inferior relationship with each. In
593 cooperation with other actors and constrained by the messages exchanged with the Inferior,
594 the Superior determines the **Outcome** applicable to the Inferior and informs the Inferior by
595 sending CONFIRM or CANCEL. This outcome can be confirm only if a PREPARED
596 message is received from the Inferior, and if a record, identifying the Inferior can be
597 persisted. (Whether this record is also a record of a confirm decision depends on the
598 Superior’s position in the business transaction as a whole.). The Superior must retain this
599 persistent record until it receives a CONFIRMED (or, in exceptional cases, CANCELLED or
600 HAZARD) from the Inferior.
601

602
603 A Superior may delegate the taking of the confirm or cancel decision to an Inferior, if there is
604 only one Inferior, by sending CONFIRM_ONE_PHASE.
605

606 A Superior may be *Atomic* or *Cohesive*; an Atomic Superior will apply the same decision to
607 all of its Inferiors; a Cohesive Superior may apply confirm to some Inferiors and cancel to
608 others, or may confirm some after others have reported cancellation. The set of Inferiors that
609 the Superior confirms (or attempts to confirm) is called the “confirm-set”.
610

611 If RESIGN is received from an Inferior, the Superior:Inferior relationship is ended; the
612 Inferior has no further effect on the behaviour of the Superior as a whole.
613

614 A Superior receives

615
616 ENROL

617
618 to enrol a new Inferior, establishing a new Superior:Inferior relationship.
619

620 A Superior sends
621

622 ENROLLED

623

624 in reply to ENROL, if the appropriate parameter on the ENROL asked for the reply.

625

626 A Superior sends

627

628 PREPARE

629 CONFIRM

630 CANCEL

631 RESIGNED

632 CONFIRM_ONE_PHASE

633 SUPERIOR_STATE

634

635 to an enrolled Inferior.

636

637 A Superior receives

638

639 PREPARED

640 CANCELLED

641 CONFIRMED

642 HAZARD

643 RESIGN

644 INFERIOR_STATE

645

646 from an enrolled Inferior.

647

648 **Inferior**

649

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

652

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

659

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

664

665 An Inferior that has become prepared may, exceptionally, make an autonomous decision to be
666 applied to the associated operations, without waiting for the Outcome from the Superior. It is
667 required to persist this autonomous decision and report it to the Superior with CONFIRMED
668 or CANCELLED as appropriate. If, when CONFIRM or CANCEL is received, the

669 autonomous decision and the decision received from the Superior are contradictory, the
670 Inferior must retain the record of the autonomous decision until receiving a
671 CONTRADICTION message.

672 An Inferior receives

673
674 PREPARE
675 CONFIRM
676 CANCEL
677 RESIGNED
678 CONFIRM_ONE_PHASE
679 SUPERIOR_STATE
680

681 from its Superior.

682 An Inferior sends

683
684 PREPARED
685 CANCELLED
686 CONFIRMED
687 HAZARD
688 RESIGN
689 INFERIOR_STATE
690

691 to its Superior.

692
693
694 ~~An Inferior receives REQUEST_STATUS and replies with STATUS. If it is also a Superior,~~
695 ~~the STATUS concerns the Inferior as a whole.~~

696 Enroller

697
698 Causes the enrolment of an Inferior with a Superior. This role is distinguished because in
699 some implementations the enrolment request will be performed by the application, in some
700 the application will ask the actor that will play the role of Inferior to enrol itself, and a
701 Factory may enrol a new Inferior (which will also be Superior) as a result of receiving
702 BEGIN&CONTEXT.
703

704 An Enroller sends

705
706 ENROL
707

708 to a Superior.

709 An Enroller receives

710
711 ENROLLED
712

713
714
715

716 in reply to ENROL if the Enroller asked for a response when the ENROL was sent.

717

718 An ENROL message sent from an Enroller that did not require an ENROLLED response may
719 be modified *en route* to the Superior by an intermediate actor to ask for an ENROLLED
720 response to be sent to the intermediate. (This may occur in the “one-shot” scenario, where an
721 ENROL/no-rsp-req is received in relation to a CONTEXT_REPLY/related; the receiver of
722 the CONTEXT_REPLY will need to ensure the enrolment is successful).

723

724 **Participant**

725

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

731

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

735

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

740

Note – Possible approaches are:

741

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

745

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

748

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

751

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

753

754 **Sub-coordinator**

755

756 An Inferior which is also an Atomic Superior.

757

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

760
761 From the perspective of its Superior (the one the sub-coordinator is Inferior to), there is no
762 difference between a sub-coordinator and any other Inferior. From this perspective, the
763 “associated operations” of the sub-coordinator as an Inferior include the relationships with its
764 Inferiors.

765
766 A sub-coordinator does not become prepared (and send PREPARED to its Superior) until and
767 unless it has received PREPARED (or RESIGN) from all its Inferiors. The outcome is
768 propagated to all Inferiors.

769 **Sub-composer**

770
771 An Inferior which is also a Cohesive Superior.

772
773
774 Like a sub-coordinator, a sub-composer cannot be distinguished from any other Inferior from
775 the perspective of its Superior.

776
777 A sub-composer is similar to a sub-coordinator, except that the constraints linking the
778 different Inferiors concern only those Inferiors in the confirm-set. How the confirm-set is
779 controlled, and when, is not defined in this specification.

780
781 If the sub-composer is instructed to cancel, by receiving a CANCEL message from its
782 Superior, the cancellation is propagated to all its Inferiors.

783
784

785 **Roles involved in the Terminator:Decider control relationships**

786

787 **Decider**

788

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

795

796 A Decider is instructed to cancel by receiving CANCEL TRANSACTION ~~whole~~.

797

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

801

802 All Deciders receive

803 REQUEST_CONFIRM CONFIRM_TRANSACTION

804 REQUEST_CANCEL TRANSACTION ~~whole~~

805 ~~REQUEST_STATUSES~~ REQUEST INFERIOR STATUSES |

806

807 All Deciders send

808 ~~CONFIRMED~~ COMPLETE |

809 ~~CANCELLED~~ COMPLETE |

810 INFERIOR_STATUSES

811

812 ~~A Decider also receives REQUEST_STATUS and replies with DECIDER_STATUS,~~ |

813 ~~reporting its state as a whole.~~ |

814

815 **Coordinator**

816

817 A Decider that is an Atomic Superior. The same outcome decision will be applied to all

818 Inferiors (excluding any from which RESIGN is received).

819

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

821

822 A Coordinator must make a cancel decision if

823 it is instructed to cancel by the Terminator

824 if CANCELLED is received from any Inferior

825 if it is unable to persist a confirm decision

826

827 **Composer**

828

829 A Decider that is a Cohesive Superior. If the Terminator requests confirmation of the

830 Cohesion, that request will determine the confirm-set of the Cohesion.

831

832 PREPARED must be received from all Inferiors in the confirm-set (excluding any from

833 which RESIGN is received) for a confirm decision to be taken.

834

835 A Composer must make a cancel decision (applying to all Inferiors) if

836 it is instructed to cancel by the Terminator

837 if CANCELLED is received from any Inferior in the confirm-set

838 if it is unable to persist a confirm decision

839

840 A Composer may be asked to prepare some or all of its Inferiors by receiving

841 ~~REQUEST_PREPARE~~ PREPARE INFERIORS. It issues PREPARE to any of those

842 Inferiors from which none of PREPARED, CANCELLED or RESIGNED have been

843 received, and replies to the ~~REQUEST_PREPARE~~ PREPARE INFERIORS with

844 INFERIOR_STATUSES.

845

846 A Composer may be asked to cancel some of its Inferiors, but not itself, by receiving

847 ~~REQUEST_CANCEL~~ INFERIORS/inferiors.

848

849 ~~In addition to the messages received by the Composer as a Decider, it receives~~ |

850 ~~— REQUEST_PREPARE~~ |

851 ~~— REQUEST_CANCEL/inferiors~~ |

852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898

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~~ CONFIRM_TRANSACTION to the target Decider. If the Decider is a Cohesion Composer, the Terminator may select which of the Composer's Inferiors are to be included in the confirm-set. If the Decider is an Atom Coordinator, all Inferiors are included. After applying the decision, the Decider replies with ~~CONFIRMED~~ COMPLETE, ~~CANCELLED~~ COMPLETE or (in the case of problems) INFERIOR_STATUSES.

A Terminator may ask a Composer (but not a Coordinator) to prepare some or all of its Inferiors with ~~REQUEST_PREPARE~~ PREPARE_INFERIORS/~~inferiors~~. The Composer replies with INFERIOR_STATUSES.

A Terminator may send ~~REQUEST_CANCEL~~ TRANSACTION to instruct the Decider to cancel the whole business transaction, ~~or, if it is a Cohesion Composer, some of its Inferiors.~~ The Decider replies with ~~CANCELLED~~ COMPLETE if all Inferiors cancel successfully, and with INFERIOR_STATUSES in the case of problems. ~~-. If the Decider is a Cohesion Composer, the Terminator may send CANCEL_INFERIORS to cancel some of the Inferiors; the Decider always replies with~~ ~~or for a selective cancel or in the case of problems;~~ INFERIOR_STATUSES.

A Terminator may check the status of the Inferiors of the Decider by sending ~~REQUEST_STATUSES~~ REQUEST_INFERIOR_STATUSES. The Decider replies with INFERIOR_STATUSES.

A Terminator sends
~~REQUEST_CONFIRM~~ CONFIRM_TRANSACTION
~~REQUEST_CANCEL~~ TRANSACTION
CANCEL_INFERIORS
~~REQUEST_PREPARE~~ PREPARE_INFERIORS/~~inferiors~~
~~REQUEST_STATUSES~~ REQUEST_INFERIOR_STATUSES

A Terminator receives
~~CONFIRMED~~ COMPLETE
~~CANCELLED~~ COMPLETE
INFERIOR_STATUSES
DECIDER_STATUS

Initiator

899 Requests a **Factory** to create a Superior – this will either be a Decider (representing a new
900 top-level business transaction) or a sub-coordinator or sub-composer to be the Inferior of an
901 existing business transaction.

902 An Initiator sends

903 BEGIN
904 BEGIN & CONTEXT

905 to a Factory, and receives in reply

906 BEGUN & CONTEXT

907 **Factory**

908 Creates Superiors and returns the CONTEXT for the new Superior. The following types of
909 Superior are created :

910 Decider, which is either
911 Composer or
912 Coordinator
913 Sub-composer
914 Sub-coordinator

915 A Factory receives

916 BEGIN
917 BEGIN & CONTEXT

918 and replies with

919 BEGUN & CONTEXT

920 If the BEGIN has no related CONTEXT, the Factory creates a Decider, either a Cohesion
921 Composer or an Atom Coordinator, as determined by the “superior type” parameter on the
922 BEGIN.

923 If the BEGIN has a related CONTEXT, the new Superior is also enrolled as an Inferior of the
924 Superior identified by the CONTEXT. The new Superior is thus a sub-composer or sub-
925 coordinator, as determined by the “superior type” parameter on the BEGIN.

926 **Other roles**

927 **Redirector**

946 Sends a REDIRECT message to inform any actor that an address previously supplied for
947 some other actor is no longer appropriate, and to supply a new address or set of addresses to
948 replace the old one.

949
950 A Redirector may send a REDIRECT message in response to receiving a message using the
951 old address, or may send REDIRECT at its own initiative.

952 If a Superior moves from the superior-address in its CONTEXT, or an Inferior moves from
953 the inferior-address in the ENROL message, the implementation **must** ensure that a
954 Redirector catches any inbound messages using the old address and replies with a
955 REDIRECT message giving the new address. (Note that the inbound message may itself be a
956 REDIRECT message.)

957
958 A Redirector **may** also be used to change the address of other BTP actors.

959
960 After receiving a REDIRECT message, the BTP actor **must** use the new address not the old
961 one, unless failure prevents it updating its information.

962

963 Status Requestor

964

965 Requests and receives the current status of a transaction tree node – any of an Inferior,
966 Superior or a Decider, or the current status of the nodes relationships with its Inferiors, if any.
967 The role of Status Requestor has no responsibilities – it is just a name for where the
968 REQUEST_STATUS and REQUEST INFERIOR STATUSES comes from
969 (REQUEST INFERIOR STATUSES is also issued by a Terminator to a Decider).

970

971 A Status Requestor sends

972

973 REQUEST_STATUS

974

975 REQUEST INFERIOR STATUSES

976

977 and receives

978

979 STATUS

980

981 INFERIOR STATUSES

982

983 in response.

984

985 The receiver of the request can refuse to provide the status information by replying with
986 FAULT(StatusRefused). The information returned in STATUS will always relate to the
987 transaction tree node as a whole (e.g. ~~actor concerned in its role~~ as an Inferior, even if it is
988 also a Superior).

989

988 Abstract Messages and Associated Contracts

989

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

993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039

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

- ❑ deliver messages completely and correctly, or not at all (corrupted messages will not be delivered);
- ❑ report some communication failures, but will not necessarily report all (i.e. not all message deliveries are positively acknowledged within the carrier);
- ❑ sometimes deliver successive messages in a different order than they were sent;

and

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

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

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

Addresses

All of the messages except CONTEXT and CONTEXT_REPLY have a “target address” parameter and many also have other address parameters. These latter identify the desired target of other messages in the set. In all cases, the exact value will invariably have been originally determined by the implementation that is the target or desired future target.

The detailed format of the address will depend on the particular carrier protocol, but at this abstract level is considered to have three parts. The first part, the “binding name”, identifies the binding to a particular carrier protocol – some bindings are specified in this document, others can be specified elsewhere. The second part of the address, the “binding address”, is meaningful to the carrier protocol itself, which will use it for the communication (i.e. it will permit a message to be delivered to a receiver). The third part, “additional information”, is not used or understood by the carrier protocol. The “additional information” may be a structured value.

When a message is actually transmitted, the “binding name” of the target address will identify which carrier protocol is in use and the “binding address” will identify the destination, as known to the carrier protocol. The entire binding address is considered to be “consumed” by the carrier protocol implementation. All of it may be used by the sending implementation, or some of it may be transmitted in headers, or as part of a URL in the carrier protocol, but then

1040 used or consumed by the receiving implementation of the carrier protocol to direct the BTP
1041 message to a BTP-aware entity (BTP-aware in that it is capable of interpreting the BTP
1042 messages). The “additional information” of the target address will be part of the BTP
1043 message itself and used in some way by the receiving BTP-aware entity (it could be used to
1044 route the message on to some other BTP entity). Thus, for the target address, only the
1045 “additional information” field is transmitted in the BTP message and the “additional
1046 information” is opaque to parties other than the recipient.
1047

1048 For other addresses in BTP messages, all three components will be within the message.
1049

1050 All messages that concern a particular Superior:Inferior relationship have an identifier
1051 parameter for the target side as well as the compound target address. This allows full
1052 flexibility for implementation choices – an implementation can:

- 1053
- 1054 a) Use the same binding address and additional information for multiple business
1055 transactions, using the identifier parameter to locate the relevant state
1056 information;
 - 1057 b) Use the same binding address for multiple business transactions and use the
1058 additional information to locate the information; or
 - 1059 c) Use a different binding address for each business transaction.
1060

1061 Which of these choices is used is opaque to the entity sending the message – both parts of the
1062 address and the identifier originated at the recipient of this message (and were transmitted as
1063 parameters of earlier messages in the opposite direction). In cases b) and c), the identifier is to
1064 some extent redundant, although interoperation requires that it always be present.
1065

1066 BTP recovery requires that the state information for a Superior or Inferior is accessible after
1067 failure and that the peer can distinguish between temporary inaccessibility and the permanent
1068 non-existence of the state information. As is explained in “Redirection” below, BTP provides
1069 mechanisms – having a set of BTP addresses for some parameters, and the REDIRECT
1070 message – that make this possible, even if the recovered state information is on a different
1071 address to the original one (as may be the case if case c) above is used).
1072
1073

1074 **Request/response pairs**

1075

1076 Many of the messages combine in pairs as a request and its response. However, in some cases
1077 the response message is sent without a triggering request, or as a possible response to more
1078 than one type of request. To allow for this, the abstract message set treats each message as
1079 standalone; but where a request does expect a reply, a “reply-address” parameter will be
1080 present. For any message with a reply address parameter, in the case of certain errors, a
1081 FAULT message will be sent to the reply address instead of the expected reply.
1082

1083 For messages which are specified as sent between Superior and Inferior, a FAULT message is
1084 sent to the peer.
1085

1086 Compounding messages

1087 BTP messages may be sent in combination with each other, or with other (application)
1088 messages. There are two cases:

- 1089
- 1090 a) Sending the messages together where the combination has semantic
1091 significance. One message is said to be “related to” the other – the combination
1092 is termed a “group”.-
1093
- 1094 b) Sending of the messages where the combination has no semantic significance,
1095 but is merely a convenience or optimisation. This is termed “bundling” – the
1096 combination -is termed a “bundle”.

1097

1098 The form A&B is used to refer to a combination (group) where message B is sent in relation
1099 to A (“relation” is asymmetric). The form A+B is used to refer to A and B bundled together-
1100 the transmission of the bundle "A+B" is semantically identical to the transmission of A
1101 followed by the transmission of B.

1102

1103 Only certain combinations of messages are possible in a group, and the meaning of the
1104 relation is specifically defined for each such combination in the next section. A particular
1105 group is treated as a unit for transmission – it has a single target address. This is usually that
1106 of one of the messages in the group – the specification for the group defines which.

1107

1108 A “bundle” of messages may contain both unrelated messages and groups of related
1109 messages. The only constraint on which messages and groups can be bundled is that ~~in both~~
1110 eases the ~~all-messages will~~ have the same binding address, but may have different “additional
1111 information” values. (Messages within a related group may have different addresses, where
1112 the rules of their relatedness permit this). Unless constrained by the binding, any messages or
1113 groups that are to be sent to the same binding address may be bundled – the fact that the
1114 binding addresses are the same is a necessary and sufficient condition for the sender to
1115 determine that the messages can be bundled.

1116

1117 A particular and important case of related messages is where a BTP CONTEXT message is
1118 sent related to an application message. In this case, the target of the application message
1119 defines the destination of the CONTEXT message. The receiving implementation may in fact
1120 remove the CONTEXT before delivering the application message to the application (Service)
1121 proper, but from the perspective of the sender, the two are sent to the same place.
1122 The compounding mechanisms, and the multi-part address structures, support the “one-wire”
1123 and “one-shot” communication patterns.

1124

1125 In “one-wire”, all message exchanges between two sides of a Superior:Inferior relationship,
1126 including the associated application messages, pass via the same “endpoints”. These
1127 “endpoints” may in fact be relays, routing messages on to particular actors within their
1128 domain. The onward routing will require some further addressing, but this has to be opaque to
1129 the sender. This can be achieved if the relaying endpoint ensures that all addresses for actors
1130 in its domain have the relay’s address as their binding address, and any routing information it
1131 will need in its own domain is placed in the additional information. (This may involve the
1132 relay changing addresses in messages as they pass through it on the way out). On receiving a

1133 message, it determines the within-domain destination from the received additional
1134 information (which is thus rewritten) and forwards the message appropriately. The sender is
1135 unaware of this, and merely sees addresses with the same binding address, which it is
1136 permitted to bundle. The content of the “additional information” is a matter only for the relay
1137 – it could put an entire BTP address in there, or other implementation-defined information.
1138 Note that a quite different one-wire implementation can be constructed where there is no
1139 relaying, but the receiving entity effectively performs all roles, using the received identifiers
1140 to locate the appropriate state.

1141
1142 “One-shot” communication makes it possible to send an application message, receive the
1143 application reply, enrol an Inferior to be responsible for the confirm/cancel of the operations
1144 of those message and inform the Superior that the Inferior is prepared, all in one two-way
1145 exchange across the network (e.g. one request/reply of a carrier protocol). ~~concerns the~~
1146 ~~bundling of application messages, especially where the application uses a request/response~~
1147 ~~paradigm.~~ The application request is sent with a related CONTEXT message. The application
1148 response is sent with a ~~related~~ relation group of CONTEXT_REPLY/related, ~~with an~~
1149 ENROL/no-rsp-req message and a ~~related~~ PREPARED message ~~(assuming the operations~~
1150 ~~succeeded and the Inferior has decided to be prepared).~~ This is possible even if the Superior
1151 address is different from the address of the application element that sends the original
1152 message (if the application exchange is request/reply, there may not even be an identifiable
1153 address for the application element). The target addresses of the ENROL and PREPARED
1154 (the Superior address) are not transmitted; the actor that was originally responsible for adding
1155 the CONTEXT to the outbound application message remembers the Superior address and
1156 forwards the ENROL and PREPARED appropriately. ~~must have a binding address that is the~~
1157 ~~same as the target address of the application response (i.e. the reply address for the client, as~~
1158 ~~perceived by the Service)—otherwise the Service cannot determine that it should bundle the~~
1159 ~~messages together. One shot is thus a specialization of one-wire.~~

1160
1161 With “one-shot”, if there are multiple Inferiors created as a result of a single application
1162 message, there is an ENROL and PREPARED message for each sent related to the
1163 CONTEXT_REPLY, ~~with the application response and the CONTEXT_REPLY.~~ If an
1164 operation fails, a CANCELLED message ~~can be~~ is sent ~~with the response~~ instead of a
1165 PREPARED.

1166
1167 If the CONTEXT has “superior-type” of “atom”, then ~~if~~ subsequent messages to the same
1168 Service, with the same related CONTEXT/atom, can have their associated operations put
1169 under the control of the same Inferior, and only a CONTEXT_REPLY/completed is sent back
1170 with the response (if the new operations fail, it will be necessary to send back
1171 CONTEXT_REPLY/repudiated, or send CANCELLED). If the “superior type” on the
1172 CONTEXT is “cohesive”, each operation will require separate enrolment.

1173
1174 Whether the “one-shot” mechanism is used is determined by the implementation on the
1175 responding (Inferior) side. This may be subject to configuration and may also be constrained
1176 by the application or by the binding in use.

1177 ~~Where does that last bit on one-shot, one-wire belong. It needs to be in somewhere.~~
1178 ~~prf~~

1179

1180 **Extensibility**

1181
1182 To simplify interoperation between implementations of this edition of BTP with
1183 implementations of future editions, the “must-be-understood” sub-parameter as specified for
1184 Qualifiers may be defined for use with any parameter added to an existing message in a future
1185 revision of this specification. The default for “must-be-understood” shall be “true”, so an
1186 implementation receiving an unrecognised parameter without a “false” value for “must-be-
1187 understood” shall not accept it (the FAULT value “UnrecognisedParameter” is available, but
1188 other errors, including lower-layer parsing/unmarshalling errors may be reported instead). If
1189 “must-be-understood” with the value “false” is present as a sub-parameter of a parameter in
1190 any message, a receiving implementation **should** ignore the parameter.

1191
1192 How the sub-parameter is associated with the new parameter is determined by the particular
1193 binding.

1194
1195 No special mechanism is provided to allow for the introduction of completely new messages.
1196

1197 **Inferior handle**

1198
1199 Some of the messages exchanged between a Terminator and a Decider are concerned with the
1200 individual Inferiors enrolled with the Decider, and not with the business transaction as a
1201 whole. These messages distinguish the Inferiors of Decider using an “inferior handle”. This is
1202 created by the Decider and is unambiguous within the scope of the Decider .

1203
1204 The “inferior handle” is distinct from the “inferior identifier” passed on an ENROL message
1205 (among other places). The latter is created by the Inferior (or its enroller) and is required to be
1206 unambiguous within the scope of the address-as-inferior on the ENROL (and unambiguous
1207 within **any** of the individual addresses in that set of BTP addresses - the identifier must
1208 identify the Inferior across all the places it might migrate to or that have recovery
1209 responsibility for it).

1210
1211 The “inferior handle” is only used by the Terminator to refer to the inferiors of the Decider.
1212 In messages between the Decider and its Inferiors, the address-as-inferior and inferior
1213 identifier are used.

1214

1215 **Messages**

1216

1217 **Qualifiers**

1218

1219 All messages have a Qualifiers parameter which contains zero or more Qualifier values. A
1220 Qualifier has sub-parameters:

1221

Sub-parameter	Type
qualifier name	string
qualifier group	URI

must-be-understood	Boolean
to-be-propagated	Boolean
content	Arbitrary – depends on type

1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250

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

Qualifier name this identifies the meaning and use of the Qualifier, using a name that is unambiguous within the scope of the Qualifier group.

Must-be-understood if this has the value “true” and the receiving entity does not recognise the Qualifier type (or does not implement the necessary functionality), a FAULT “UnsupportedQualifier” shall be returned and the message shall not be processed. Default is “true”.

To-be-propagated if this has the value “true” and the receiving entity passes the BTP message (which may be a CONTEXT, but can be other messages) onwards to other entities, the same Qualifier value shall be included. If the value is “false”, the Qualifier shall not be automatically included if the BTP message is passed onwards. (If the receiving entity does support the qualifier type, it is possible a propagated message may contain another instance of the same type, even with the same Content – this is not considered propagation of the original qualifier.). Default is “false”.

Content the type (which may be structured) and meaning of the content is defined by the specification of the Qualifier.

Messages involved in several relationships not restricted to outcome or control relationships.

The messages in this section are used between various roles. The CONTEXT message is used in the Initiator:Factory relationship (when it is related to BEGIN or to BEGUN), and related to an application ‘message’ to propagate the business transaction between parts of the application and used on an Application:Application relationship. Another use is when it is related to a BEGUN message on an Initiator: Factory relationship. A CONTEXT REPLY is used as the reply to a CONTEXT related to an application ‘message’ and used on an Application:Application relationship. A REQUEST STATUS can be issued to, and STATUS returned by any of Decider, Superior or Inferior. FAULT message can be used on any relationship to indicate an error condition back to the sender of a message, except on an

1263 Application: Application relationship where it is assumed that the application protocol will
1264 have its own means of coping with errors.

1265

1266 CONTEXT

1267

1268 A CONTEXT is supplied by (or on behalf of) a Superior and related to one or more
1269 application messages. (The means by which this relationship is represented is determined by
1270 the binding and the binding mechanisms of the application protocol.) The “superior type”
1271 parameter identifies whether the Superior will apply the same decision to all Inferiors
1272 enrolled using the same superior identifier (“superior type” is “atom”) or whether it may
1273 apply different decisions (“superior type” is “cohesion”).
1274

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

1275

1276

1277 **address-as-superior** the address to which ENROL and other messages from an
1278 enrolled Inferior are to be sent. This can be a set of alternative addresses.

1279

1280 **superior identifier** identifies the Superior within the scope of the address-as-
1281 superior

1282

1283 reply-address the address to which a replying CONTEXT REPLY is to be sent.
1284 This may be different each time the CONTEXT is transmitted – it refers to the
1285 destination of a replying CONTEXT REPLY for this particular transmission of
1286 the CONTEXT.

1287

1288 **superior type** identifies whether the CONTEXT refers to a Cohesion or an
1289 Atom. Default is atom.

1290

1291

1292 **qualifiers** standardised or other qualifiers. The standard qualifier “Transaction
1293 timelimit” is carried by CONTEXT.

1294

1295 There is no target address parameter for CONTEXT as it is only transmitted in relation to the
1296 application messages, BEGIN and BEGUN.

1297

1298 The forms CONTEXT/cohesion and CONTEXT/atom refer to CONTEXT messages with the
1299 superior type with the appropriate value.

1300

1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311

CONTEXT_REPLY

CONTEXT_REPLY is sent after receipt of CONTEXT (related to application message(s)) to indicate whether all necessary enrolments have already completed (ENROLLED has been received) or will be completed by ENROL messages sent in relation to the CONTEXT_REPLY or if an enrolment attempt has failed. CONTEXT_REPLY may be sent related to an application message (typically the response to the application message related to the CONTEXT). In some bindings the CONTEXT_REPLY may be implicit in the application message.

Parameter	Type
<u>target-address</u>	<u>BTP address</u>
superior-address	BTP address
superior identifier	Identifier
completion_status	complete/related/repudiated
Qualifiers	List of qualifiers

1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324

target-address the address to which the CONTEXT_REPLY is sent. This shall be the "reply-address" from the CONTEXT.

superior-address one of the addresses from the address-as-superior from the CONTEXT. (The parameter is present in CONTEXT_REPLY to disambiguate the superior identifier.)

superior identifier the superior identifier from the CONTEXT

completion_status: reports whether all enrol operations made necessary by the receipt of the earlier CONTEXT message have completed. Values are

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

1325
1326
1327
1328
1329

qualifiers standardised or other qualifiers.

The form CONTEXT_REPLY/completed, CONTEXT_REPLY/related and CONTEXT_REPLY/repudiated refer to CONTEXT_REPLY messages with status having the

1330 appropriate value. The form CONTEXT_REPLY/ok refers to either of
 1331 CONTEXT_REPLY/completed or CONTEXT_REPLY/related.
 1332
 1333 If there are no necessary enrolments (e.g. the application messages related to the received
 1334 CONTEXT did not require the enrolment of any Inferiors), then
 1335 CONTEXT_REPLY/completed is used.
 1336
 1337 If a CONTEXT_REPLY/repudiated is received, the receiving implementation **must** ensure
 1338 that the business transaction will not be confirmed.
 1339
 1340

1341 REQUEST_STATUS

1342
 1343 Sent to an Inferior, Superior or to a Decider to ask it to reply with STATUS. The receiver
 1344 may reject the request with a FAULT(StatusRefused).
 1345

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

1346
 1347 **target address** the address to which the REQUEST_STATUS message is sent.
 1348 This can be any of address-as-decider, address-as-inferior or address-as-
 1349 superior. If the target is an Inferior, this will be the address-as-inferior on the
 1350 ENROL message. If the target is a Decider, this will be the address-as-decider on
 1351 the BEGUN message.
 1352

1353 **reply address** the address to which the replying STATUS should be sent.
 1354

1355 ~~inferior-target~~ identifier The identifier for the business transaction, or part of
 1356 business transaction whose status is sought. If the target-address is an address-as-
 1357 decider, this parameter shall be the “transaction-identifier” on the BEGUN
 1358 message. If the target-address is an address-as-inferior, this parameter shall be
 1359 an Inferior, the “inferior-identifier” on the ENROL message. If the target-address
 1360 is a an address-as-superior, this parameter shall be the “superior-identifier” on the
 1361 CONTEXT. Decider, this parameter shall be absent.
 1362

1363 ~~transaction-identifier~~ If the target is a Decider, the “transaction-identifier” on
 1364 the BEGUN message. If the target is an Inferior, this parameter shall be absent.
 1365

1366 **qualifiers** standardised or other qualifiers.
 1367

1368 Types of FAULT possible (sent to reply address)

1369

1370

1371

1372

1373

1374

1375

1376 STATUS

1377

1378

1379

1380

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

Parameter	Type
target address	BTP address
<u>responders</u> address-as-inferior	BTP address
<u>inferior</u> <u>responders</u> -identifier	Identifier
status	See below
qualifiers	List of qualifiers

1381

1382

1383

1384

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

1385

1386

1387

1388

1389

1390

1391

1392

1393

respondersaddress-as-inferior the address of the sender of the STATUS message – one of address-as-inferior, address-as-decider, address-as-superior(with the responders-identifier, this determines who the message is from).. If the sender has different addresses as multiple roles (as Decider, Inferior or Superior), this shall be the address on which the REQUEST_STATUS was received. If the sender is an Inferior, the address as inferior as on the ENROL message (with the inferior-identifier, this determines who the message is from). If the sender is a Decider, this parameter shall be absent

1394

1395

1396

1397

1398

1399

respondersinferior-identifier the identifier of the state, aligned with the responders-address. If the sender has multiple roles in the transaction (as Decider, Inferior or Superior), this shall be the target-identifier on the REQUEST_STATUS. If the sender is an Inferior, the inferior-identifier as on the ENROL message. If the sender is a Decider, this parameter shall be absent.

1400

1401

1402

1403

~~address-as-decider~~ If the sender is a Decider, the address-as-decider on the BEGUN message (with the “transaction-identifier”, this determines who the message is from). If the sender is an Inferior, this parameter shall be absent.

1404

1405

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

1406
 1407
 1408
 1409
 1410
 1411
 1412

status states the current status of the transaction tree node represented by the sender. Some of the values are only issued if the sender is an Inferior. If the transaction tree node is both Superior and Inferior (i.e. is a sub-coordinator or sub-composer), and two status values would be valid for the current state, it is the sender's option which one is used.

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

status value	Meaning from Decider <u>Superior</u>	Meaning from Inferior
<i>contradiction</i>		was made, CANCEL received; CONTRADICTION has not been received
<i>Hazard</i>	A hazard has been reported from at least one Inferior	A hazard has been discovered; CONTRADICTION has not been received
<i>Contradicted</i>	Not applicable	CONTRADICTION has been received
<i>Unknown</i>	No state information for the transaction <u>target</u> identifier exists; no such Decider exists	No state information for the <u>target</u> identifier exists; no such Inferior exists
<i>Inaccessible</i>	There may be state information for this <u>target</u> identifier but it cannot be reached/existence cannot be determined	There may be state information for this <u>target</u> identifier but it cannot be reached/existence cannot be determined

1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425

qualifiers standardised or other qualifiers.

Types of FAULT possible (~~sent to address as decider~~)

General

~~Invalid Terminator~~ — if Terminator address is unknown

~~Unknown Transaction~~ — if the transaction identifier is unknown

FAULT

Sent in reply to various messages to report an error condition

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

1426
1427
1428
1429
1430

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

1431 superior identifier the superior identifier as on the CONTEXT message and as
1432 used on the ENROL message (present only if the FAULT is sent to the superior).
1433
1434 inferior identifier the inferior identifier as on the ENROL message (present only
1435 if the FAULT is sent to the inferior)
1436
1437 fault type identifies the nature of the error, as specified for each of the main
1438 messages.
1439
1440 fault data information relevant to the particular error. Each fault type defines the
1441 content of the fault data:
1442

1443

<u>fault type</u>	<u>meaning</u>	<u>fault data</u>
<u>CommunicationFailure</u>	<u>Any fault arising from the carrier mechanism and communication infrastructure.</u>	<u>Determined by the carrier mechanism and binding specification</u>
<u>DuplicateInferior</u>	<u>An inferior with the same address and identifier is already enrolled with this Superior</u>	<u>The identifier</u>
<u>General</u>	<u>Any otherwise unspecified problem</u>	<u>Free text explanation</u>
<u>InvalidDecider</u>	<u>The address the message was sent to is not valid (at all or for this Terminator and transaction identifier)</u>	<u>The address</u>
<u>InvalidInferior</u>	<u>The Superior is known but the Inferior identified by the address-as-inferior and identifier are not enrolled in it</u>	<u>The Inferior Identity (address-as-inferior and identifier)</u>
<u>InvalidSuperior</u>	<u>The received identifier is not known or does not identify a known Superior</u>	<u>The identifier</u>
<u>StatusRefused</u>	<u>The receiver will not report the request status (or inferior statuses) to this StatusRequestor</u>	<u>Free text explanation</u>
<u>InvalidTerminator</u>	<u>The address the message was sent to is not valid (at all or for this Decider and transaction identifier)</u>	<u>The address</u>
<u>UnknownParameter</u>	<u>A BTP message has been received with an unrecognised parameter</u>	<u>Free text explanation</u>
<u>UnknownTransaction</u>	<u>The transaction-identifier is unknown</u>	<u>The transaction-identifier</u>
<u>UnsupportedQualifier</u>	<u>A qualifier has been received that is not recognised and on which "must-be-Understood" is "true".</u>	<u>Qualifier group and name</u>
<u>WrongState</u>	<u>The message has arrived when the recipient is in an invalid state.</u>	

1444

1445 UnknownParameter A BTP message has been Free text explanation
 1446 received with an unrecognised
 1447 q parameter
 1448 u
 1449 Qualifiers standardised or other qualifiers.
 1450

1451 Note – If the carrier mechanism used for the transmission of BTP messages
 1452 is capable of delivering messages in a different order than they were sent in,
 1453 the “WrongState” FAULT is not sent and should be ignored if received.

1454
 1455 REQUEST_INFERIOR_STATUSES, INFERIOR_STATUSES
 1456

1457 REQUEST_INFERIOR_STATUSES may be sent to and INFERIOR_STATUSES sent from
 1458 any Decider, Superior or Inferior, asking it to report on the status of its relationships with
 1459 Inferiors (if any). Since Deciders are required to respond to
 1460 REQUEST_INFERIOR_STATUSES with INFERIOR_STATUSES but non-Deciders may
 1461 just issue FAULT(StatusRefused), and INFERIOR_STATUSES is also used as a reply to
 1462 other messages from Terminator to Decider, these messages are described below under the
 1463 messages used in the control relationships.
 1464

1465 Messages involved used in the Superior:Inferior outcome relationships
 1466

1467 **ENROL**
 1468

1469 A request to a Superior to ENROL an Inferior. This is typically issued after receipt of a
 1470 CONTEXT message in relation to an application request.

1471 The actor issuing ENROL plays the role of Enroller.
 1472

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

1473
 1474 **target address** the address to which the ENROL is sent. This will be the
 1475 address-as-superior from the CONTEXT message.
 1476

1477 **superior identifier**. The superior identifier as on the CONTEXT message
1478
1479 **reply requested** true if an ENROLLED response is required, false otherwise.
1480 Default is false.
1481
1482 **reply address** the address to which a replying ENROLLED is to be sent, if
1483 “reply requested” is true. If this field is absent and “reply requested” is true, the
1484 ENROLLED should be sent to the “address-as-inferior” (or one of them, at
1485 sender’s option)
1486
1487 **address-as-inferior** the address to which PREPARE, CONFIRM, CANCEL and
1488 SUPERIOR_STATE messages for this Inferior are to be sent.
1489
1490 **inferior identifier** an identifier that unambiguously identifies this Inferior within
1491 the scope of any of the address-as-inferior set of BTP-addresses.
1492
1493 **qualifiers** standardised or other qualifiers. The standard qualifier “Inferior
1494 name” may be present.
1495

1496 Types of FAULT possible (sent to Reply address)

1497
1498 **General**

1499 **InvalidSuperior** – if superior identifier is unknown

1500 **DuplicateInferior** – if inferior with at least one of the set address-as-
1501 inferior the same and the same inferior identifier is already enrolled

1502 **WrongState** – if it is too late to enrol new Inferiors (generally if the
1503 Superior has already sent a PREPARED message to its superior or
1504 terminator, or if it has already issued CONFIRM to other Inferiors).
1505

1506 The form ENROL/rsp-req refers to an ENROL message with “reply requested” having the
1507 value “true”; ENROL/no-rsp-req refers to an ENROL message with “reply requested” having
1508 the value “false”
1509

1510 ENROL/no-rsp-req is typically sent in relation to CONTEXT_REPLY/related. ENROL/rsp-
1511 req is typically when CONTEXT_REPLY/completed will be used (after the ENROLLED
1512 message has been received.)
1513

1514 **ENROLLED**

1515
1516 Sent from Superior in reply to an ENROL/rsp-req message, to indicate the Inferior has been
1517 successfully enrolled (and will therefore be included in the termination exchanges)
1518

Parameter	Type
target address	BTP address
inferior identifier	Identifier

Parameter	Type
inferior-handle	Handle
Qualifiers	List of qualifiers

1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545

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

inferior identifier The inferior identifier as on the ENROL message

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

qualifiers standardised or other qualifiers.

No FAULT messages are issued on receiving ENROLLED.

RESIGN

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

RESIGN may be sent at any time prior to the sending of a PREPARED or CANCELLED message (which cannot then be sent). RESIGN may be sent in response to a PREPARE message.

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

1546
1547
1548
1549
1550

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

superior-identifier The superior identifier as on the ENROL message

1551
 1552 **address-as-inferior** The address-as-inferior as on the earlier ENROL message
 1553 (with the inferior identifier, this determines who the message is from)
 1554
 1555 **inferior-identifier** The inferior identifier as on the earlier ENROL message
 1556
 1557 **response-requested** is set to “true” if a RESIGNED response is required.
 1558
 1559 **qualifiers** standardised or other qualifiers.

1560
 1561 Note -- RESIGN is equivalent to readonly vote in some other protocols, but can be issued
 1562 early.

1563
 1564 Types of FAULT possible (sent to address-as-inferior)

1565
 1566 *General*
 1567 *InvalidSuperior* – if superior identifier is unknown
 1568 *InvalidInferior* – if no ENROL had been received for this address-as-
 1569 inferior and identifier (Inferior Identity)
 1570 *WrongState* – if a PREPARED or CANCELLED has already been
 1571 received by the Superior from this Inferior
 1572

1573 The form RESIGN/rsp-req refers to an RESIGN message with “reply requested” having the
 1574 value “true”; RESIGN /no-rsp-req refers to an RESIGN message with “reply requested”
 1575 having the value “false”
 1576

1577
 1578 **RESIGNED**

1579 Sent in reply to a RESIGN/rsp-req message.
 1580
 1581

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

1582
 1583 **target address** the address to which the RESIGNED is sent. This will be the
 1584 address-as-inferior from the ENROL message.
 1585

1586 **inferior identifier** The inferior identifier as on the earlier ENROL message for
 1587 this Inferior.
 1588

1589 **qualifiers** standardised or other qualifiers.
 1590

1591 After receiving this message the Inferior will not receive any more messages with this
1592 address-as-inferior and identifier.

1593
1594 No FAULT messages are issued on receiving RESIGNED.
1595

1596 PREPARE

1597
1598 Sent from Superior to an Inferior from whom ENROL but neither CANCELLED nor
1599 RESIGN have been received, requesting a PREPARED message. PREPARE can be sent after
1600 receiving a PREPARED message.
1601

1602 ~~Sent from a Terminator to a Composer to tell it to prepare all or some of its inferiors, by~~
1603 ~~sending PREPARE to any that have not already sent PREPARED, RESIGN or~~
1604 ~~CANCELLED to the Composer. If the inferiors-list parameter is absent, the request applies to~~
1605 ~~all the inferiors; if the parameter is present, it applies only to the identified inferiors of the~~
1606 ~~Composer.~~
1607

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

1608
1609 **target address** the address to which the PREPARE message is sent. When sent
1610 from Superior to Inferior, this will be the address-as-inferior from the ENROL
1611 message. ~~When sent from Terminator to Composer, this will be the decider-~~
1612 ~~address from the BEGUN message.~~

1613
1614 **inferior identifier** When sent from Superior to Inferior, the inferior identifier as
1615 on the earlier ENROL message. ~~This parameter shall be absent when sent from~~
1616 ~~Terminator to Composer.~~

1617
1618 ~~**reply address** When sent from Terminator to Composer, the address of the~~
1619 ~~Terminator sending the PREPARE message. This parameter shall be absent when~~
1620 ~~sent from Superior to Inferior.~~

1621
1622 ~~**transaction identifier** When sent from Terminator to Composer, identifies the~~
1623 ~~Composer and will be the transaction identifier from the BEGUN message.. This~~
1624 ~~parameter shall be absent when sent from Superior to Inferior.~~

1625
1626 ~~**inferiors list** When sent from Terminator to Composer, defines which of the~~
1627 ~~inferiors of this Composer preparation is requested for. If this parameter is absent~~

1628 ~~when sent to a Composer, the PREPARE applies to all Inferiors. This parameter~~
1629 ~~shall be absent when sent from Superior to Inferior.~~

1630

1631 **qualifiers** standardised or other qualifiers. The standard qualifier “Minimal
1632 inferior timeout” is carried by PREPARE.

1633

1634

1635 On receiving PREPARE, an Inferior **should** reply with a PREPARED, CANCELLED or
1636 RESIGN.

1637

1638 ~~When sent to a Composer, for all Inferiors identified in the inferiors-list parameter (all~~
1639 ~~Inferiors if the parameter is absent), from which none of PREPARED, CANCELLED or~~
1640 ~~RESIGNED has been received, the Composer shall issue PREPARE. It will reply to the~~
1641 ~~Terminator, using the reply address on the PREPARE message, sending an~~
1642 ~~INFERIOR_STATUSES message giving the status of the Inferiors identified on the inferiors-~~
1643 ~~list parameter (all of them if the parameter was absent).~~

1644

1645 Types of FAULT possible (sent to Superior address)

1646

1647 **General**

1648 ~~UnknownTransaction~~ – if the transaction identifier is unknown

1649 **InvalidInferior** – if inferior identifier is unknown, or an inferior-handle
1650 on the inferiors-list is unknown

1651 **WrongState** – if a CONFIRM or CANCEL has already been received by
1652 this Inferior; ~~if a REQUEST_CONFIRM or CANCEL/whole has already~~
1653 ~~been received by this Composer.~~

1654

1655 ~~The form PREPARE/whole refers to a PREPARE message sent to a Composer where the~~
1656 ~~“inferiors list” parameter is absent. The form PREPARE/inferiors refers to a PREPARE~~
1657 ~~message sent to a Composer where the “inferiors list” parameter is present. The unqualified~~
1658 ~~form PREPARE is used for a PREPARE message sent to an Inferior.~~

1659

1660 PREPARED

1661

1662 Sent from Inferior to Superior, either unsolicited or in response to PREPARE, but only when
1663 the Inferior has determined the operations associated with the Inferior can be confirmed and
1664 can be cancelled, as may be instructed by the Superior. The level of isolation is a localmatter
1665 (i.e. it is the Inferiors choice, as constrained by the shared understanding of the application
1666 exchanges) – other access may be blocked, may see applied results of operations or may see
1667 the original state.

1668

Parameter	Type
target address	BTP address
superior identifier	Identifier
address-as-inferior	Set of BTP addresses

inferior identifier	Identifier
default is cancel	Boolean
qualifiers	List of qualifiers

1669

1670

1671

1672

1673

1674

1675

1676

1677

1678

1679

1680

1681

1682

1683

1684

1685

1686

1687

1688

1689

1690

1691

1692

1693

1694

1695

1696

1697

1698

1699

1700

1701

1702

1703

1704

1705

1706

1707

1708

1709

1710

target address the address to which the PREPARED is sent. This will be the Superior address as on the ENROL message.

superior identifier When the message is sent from an Inferior to the Superior, the superior identifier as on the ENROL message

address-as-inferior When the message is sent from an Inferior to the Superior, the address-as-inferior as on the earlier ENROL message (with the inferior identifier, this determines who the message is from)

inferior identifier The inferior identifier as on the ENROL message

default is cancel if “true”, the Inferior states that if the outcome at the Superior is to cancel the operations associated with this Inferior, no further messages need be sent to the Inferior. If the Inferior does not receive a CONFIRM message, it will cancel the associated operations. The value “true” will invariably be used with a qualifier indicating under what circumstances (usually a timeout) an autonomous decision to cancel will be made. If “false”, the Inferior will expect a CONFIRM or CANCEL message as appropriate, even if qualifiers indicate that an autonomous decision will be made.

qualifiers standardised or other qualifiers. The standard qualifier “Inferior timeout” may be carried by PREPARED.

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.

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

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”.

1711
1712
1713
1714
1715

CONFIRM

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

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

1716
1717
1718
1719

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

1720
1721
1722

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

1723

qualifiers standardised or other qualifiers.

1724

1725

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

1726

1727

1728

Types of FAULT possible (sent to Superior address)

1729

1730

General

1731

InvalidInferior – if inferior identifier is unknown

1732

WrongState – if no PREPARED has been sent by, or if CANCEL has been received by this Inferior.

1733

1734

1735

1736

CONFIRMED

1737

1738

Sent after the Inferior has applied the confirmation, both in reply to CONFIRM or when the Inferior has made an autonomous confirm decision, and in reply to a

1740

CONFIRM_ONE_PHASE if the Inferior decides to confirm its associated operations.

1741

1742

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

1743

1744

1745

1746

Parameter	Type
target address	BTP address
superior identifier	Identifier

Parameter	Type
address-as-inferior	Set of BTP addresses
inferior identifier	Identifier
address-as-decider	BTP address
transaction identifier	identifier
confirm received	Boolean
qualifiers	List of qualifiers

1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783

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.~~

1784 **qualifiers** standardised or other qualifiers.

1785

1786 Types of FAULT possible (sent to address-as-inferior)

1787

1788 **General**

1789 **InvalidSuperior** – if Superior identifier is unknown

1790 **InvalidInferior** – if no ENROL has been received for this address-as-
1791 inferior and identifier, or if RESIGN has been received from this Inferior.

1792

1793 Note – A CONFIRMED message arriving before a CONFIRM message is
1794 sent, or after a CANCEL 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 The form CONFIRMED/auto refers to a CONFIRMED message with “confirm
1799 received” = “false”; CONFIRMED/response refers to a CONFIRMED message
1800 with “confirm received” = ”true”. ~~The unqualified form CONFIRMED refers to~~
1801 ~~the message without an confirm received parameter, as used between Decider~~
1802 ~~and Terminator.~~

1803

1804

1805 **CANCEL**

1806

1807 Sent by the Superior to an Inferior at any time before (and unless) CONFIRM has been sent.

1808

1809 ~~Sent by a Terminator to a Decider at any time before REQUEST_CONFIRM has been sent.~~

1810

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

1811

1812 **target address** the address to which the CANCEL message is sent. When sent
1813 from Superior to Inferior, this will be the address-as-inferior from the ENROL
1814 message. ~~When sent from Terminator to Composer, this will be the decider~~
1815 ~~address from the BEGUN message.~~

1816

1817 inferior identifier When sent from Superior to Inferior, the inferior identifier as
1818 on the earlier ENROL message. ~~This parameter shall be absent when sent from~~
1819 ~~Terminator to Decider.~~
1820
1821 ~~reply address~~ When sent from Terminator to Decider, the address of the
1822 ~~Terminator sending the CANCEL message. This parameter shall be absent when~~
1823 ~~sent from Superior to Inferior.~~
1824
1825 ~~transaction identifier~~ When sent from Terminator to Decider, identifies the
1826 ~~Decider and will be the transaction identifier from the BEGUN message.. This~~
1827 ~~parameter shall be absent when sent from Superior to Inferior.~~
1828
1829 ~~inferiors list~~ When sent from Terminator to Composer, defines which of the
1830 ~~Inferiors of this Composer are to be cancelled. This parameter shall be absent~~
1831 ~~when sent from a Superior to an Inferior and when sent from a Terminator to a~~
1832 ~~Coordinator.~~
1833
1834 **qualifiers** standardised or other qualifiers.

1835
1836 When sent to an Inferior, the effects of any operations associated with the Inferior should be
1837 undone. If the Inferior had sent PREPARED, the Inferior is released from its promise to be
1838 able to confirm the operations.

1839 ~~When sent to a Decider with the inferiors list parameter is absent, the business transaction is~~
1840 ~~cancelled — this is propagated to any remaining Inferiors by issuing CANCEL to them. No~~
1841 ~~more Inferiors will be permitted to enrol.~~

1842
1843 ~~When sent to a Composer, with the inferiors list parameter present, only the Inferiors~~
1844 ~~identified in the inferiors list are to be cancelled. Any other inferiors are unaffected by a~~
1845 ~~CANCEL/inferiors. Further Inferiors may be enrolled.~~

1846
1847

1848 ~~Note — A CANCEL/inferiors issued to a Cohesion Composer identifying all~~
1849 ~~of its currently enrolled Inferiors will leave the Cohesion ‘empty’, but~~
1850 ~~permitted to continue with new Inferiors, if any enrol.~~

1851
1852 Types of FAULT possible (sent to Superior address)

1853
1854 **General**

1855 ~~UnknownTransaction~~ if the ~~transaction identifier is unknown~~
1856 **InvalidInferior** – if inferior identifier is unknown, or an inferior-handle
1857 on the inferiors-list is unknown

1858 **WrongState** – if a CONFIRM has been received by this Inferior; ~~if a~~
1859 ~~REQUEST_CONFIRM has been received by this Composer.~~
1860

1861 ~~The form CANCEL/whole refers to a CANCEL message sent to a Decider where the~~
1862 ~~“inferiors list” parameter is absent. The form CANCEL/inferiors refers to a CANCEL~~
1863 ~~message sent to a Composer where the “inferiors-list” parameter is present. The unqualified~~
1864 ~~form CANCEL is used to refer to a CANCEL message sent from a Superior to an Inferior.~~

1867 CANCELLED

1868
1869 Sent when the Inferior has applied (or is applying) cancellation of the operations associated
1870 with the Inferior. CANCELLED is sent from Inferior to Superior in the following cases:

- 1871 1. before (and instead of) sending PREPARED, to indicate the Inferior is unable to
1872 apply the operations in full and is cancelling all of them;
- 1873 2. in reply to CANCEL, regardless of whether PREPARED has been sent;
- 1874 3. after sending PREPARED and then making and applying an autonomous
1875 decision to cancel.
1876
- 1877 4. in reply to CONFIRM_ONE_PHASE if the Inferior decides to cancel the
1878 associated operations
1879

1880
1881
1882 As is specified in the state tables, cases 1, 2 and 3 are not always distinct in some
1883 circumstances of recovery and resending of messages.

1884
1885
1886 ~~CANCELLED is also sent by Decider to a Terminator in reply to REQUEST_CONFIRM if~~
1887 ~~all Inferiors cancel without reporting hazards.~~

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

1889
1890 **target address** the address to which the CANCELLED is sent. When sent by an
1891 Inferior to a Superior, this will be the Superior address as on the CONTEXT
1892 message. ~~When sent from a Decider to a Terminator it will be the reply address~~
1893 ~~from the REQUEST_CONFIRM message.~~
1894

1895 **superior identifier** When the message is sent from an Inferior to the Superior,
1896 this shall be the superior identifier as on the CONTEXT message. ~~This parameter~~
1897 ~~shall be absent when CANCELLED is sent from Decider to Terminator.~~
1898

1899 **address-as-inferior** When the message is sent from an Inferior to the Superior,
1900 this shall be the address-as-inferior as on the earlier ENROL message (with the
1901 inferior identifier, this determines who the message is from). ~~This parameter shall~~
1902 ~~be absent when CANCELLED is sent from Decider to Terminator.~~
1903

1904 **inferior identifier** When the message is sent from an Inferior to the Superior, this
1905 shall be the inferior identifier as on the earlier ENROL message. ~~This parameter~~
1906 ~~shall be absent when CANCELLED is sent from Decider to Terminator.~~
1907

1908 **address-as-decider** When the message is sent from a Decider to the
1909 Terminator, ~~this shall be the address as decider of the Decider as on the BEGUN~~
1910 ~~message (with the transaction identifier, this determines who the message is~~
1911 ~~from). This parameter shall be absent when CANCELLED is sent from an~~
1912 ~~Inferior to Superior.~~
1913

1914 ~~**transaction identifier** When the message is sent from a Decider to the~~
1915 ~~Terminator, this shall be the transaction identifier as on the BEGUN message (i.e.~~
1916 ~~the identifier of the Decider as a whole). This parameter shall be absent when~~
1917 ~~CANCELLED is sent from an Inferior to Superior~~
1918

1919 **qualifiers** standardised or other qualifiers.

1920
1921 Types of FAULT possible (sent to address-as-inferior)
1922

1923 *General*
1924 *InvalidSuperior* – if Superior identifier is unknown
1925 *InvalidInferior* – if no ENROL has been received for this address-as-
1926 inferior and identifier, or if RESIGN has been received from this Inferior
1927 *WrongState* – if CONFIRM has been sent
1928

1929 Note – A CANCELLED message arriving before a CANCEL message is
1930 sent, or after a CONFIRM has been sent will occur when the Inferior has
1931 taken an autonomous decision and is not regarded as occurring in the wrong
1932 state. (The latter will cause a CONTRADICTION message to be sent.)

1933
1934
1935 **CONFIRM_ONE_PHASE**
1936

1937 Sent from a Superior to an enrolled Inferior, when there is only one such enrolled Inferior. In
1938 this case the two-phase exchange is not performed between the Superior and Inferior and the
1939 outcome decision for the operations associated with the Inferior is determined by the Inferior.

1940

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

1941

1942

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

1943

1944

1945

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

1946

1947

1948

report hazard Defines whether the superior wishes to be informed if a mixed condition occurs for the operations associated with the Inferior. If “report hazard” is “true”, the Inferior will reply with HAZARD if a mixed condition occurs, or if the Inferior cannot determine that a mixed condition has not occurred. If “report hazard” is false, the Inferior will report only its own decision, regardless of whether that decision was correctly and consistently applied. Default is false.

1949

1950

1951

1952

1953

1954

1955

qualifiers standardised or other qualifiers.

1956

1957

CONFIRM_ONE_PHASE can be issued by a Superior to an Inferior from whom PREPARED has been received (subject to the requirement that there is only one enrolled Inferior).

1958

1959

1960

Types of FAULT possible (sent to Superior address)

1961

1962

1963

General

1964

InvalidInferior – if inferior identifier is unknown

1965

WrongState – if a PREPARE has already been received from this

1966

Inferior

1967

HAZARD

1968

1969

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

1970

1971

1972

1973

1974

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

1975

1976

1977

1978

Parameter	Type
target address	BTP address
superior identifier	Identifier
address-as-inferior	Set of BTP addresses
inferior identifier	Identifier
level	mixed/possible
Qualifiers	List of qualifiers

1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013

target address the address to which the **MIXED HAZARD** is sent. This will be the superior address from the ENROL message.

superior identifier The superior identifier as used on the ENROL message

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

inferior identifier The inferior identifier as on the earlier ENROL message

level indicates, with value “mixed” that a mixed condition has definitely occurred; or, with value “possible” that it is unable to determine whether a mixed condition has occurred or not.

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

The form HAZARD/mixed refers to a HAZARD message with “level” = “mixed”, the form HAZARD/possible refers to a HAZARD message with “level” = “possible”.

CONTRADICTION

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

Parameter	Type
-----------	------

target address	BTP address
inferior identifier	Identifier
Qualifiers	List of qualifiers

2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041

target address the address to which the CONTRADICTION 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.

qualifiers standardised or other qualifiers.

Types of FAULT possible (sent to Superior address)

General

InvalidInferior – if inferior identifier is unknown

WrongState – if neither CONFIRMED or CANCELLED has been sent by this Inferior

SUPERIOR_STATE

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	<i>see below</i>
reply requested	Boolean
Qualifiers	List of qualifiers

2042
2043
2044
2045

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

2046 **inferior identifier** The inferior identifier as on the earlier ENROL message for
2047 this Inferior.

2048
2049 **status** states the current state of the Superior, in terms of its relation to this
2050 Inferior only.

status value	Meaning
<i>active</i>	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)
<i>prepared-received</i>	PREPARED has been received from the Inferior, but no outcome is yet available
<i>inaccessible</i>	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
<i>unknown</i>	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

2052
2053 **Reply requested** true, if SUPERIOR_STATE is sent as a query at the Superior's
2054 initiative; false, if SUPERIOR_STATE is sent in reply to a received
2055 INFERIOR_STATE or other message. Can only be true if status is active or
2056 prepared-received.

2057
2058 **qualifiers** standardised or other qualifiers.

2059
2060 The Inferior, on receiving SUPERIOR_STATE with reply requested = true, should reply in a
2061 timely manner by (depending on its state) repeating the previous message it sent or by
2062 sending INFERIOR_STATE with the appropriate status value.

2063
2064 A status of unknown shall only be sent if it has been determined for certain that the Superior
2065 has no knowledge of the Inferior, or (equivalently) it can be determined that the relationship
2066 with the Inferior was cancelled. If there could be persistent information corresponding to the
2067 Superior, but it is not accessible from the entity receiving an INFERIOR_STATE/*y (or
2068 other) message targeted to the Superior or that entity cannot determine whether any such
2069 persistent information exists or not, the response shall be Inaccessible.

2070
2071 SUPERIOR_STATE/unknown is also used as a response to messages, other than
2072 INFERIOR_STATE/*y that are received when the Inferior is not known (and it is known
2073 there is no state information for it).

2074
2075 The form SUPERIOR_STATE/abcd refers to a SUPERIOR_STATE message status having a
2076 value equivalent to "abcd" (for active, prepared-received, unknown and inaccessible) and
2077 with "reply requested" = "false". SUPERIOR_STATE/abcd/y refers to a similar message, but

2078 with “reply requested” = “true”. The form SUPERIOR_STATE/*/y refers to a
2079 SUPERIOR_STATE message with “reply requested” = “true” and any value for status.

2080
2081

2082 INFERIOR_STATE

2083

2084 Sent by an Inferior as a query when in the active state to a Superior, when (due recovery from
2085 previous failure or other reason) there is uncertainty what state the Superior has reached.

2086

2087 Also sent by the Inferior to the Superior in response to a received SUPERIOR_STATE, in
2088 particular states.

2089

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

2090

2091 **target address** the address to which the INFERIOR_STATE is sent. This will
2092 be the target address as used the original ENROL message.

2093

2094 **superior identifier** The superior identifier as used on the ENROL message

2095

2096 **address-as-inferior** The address-as-inferior as on the ENROL message (with the
2097 inferior identifier, this determines who the message is from)

2098

2099 **inferior identifier** The inferior identifier as on the ENROL message

2100

2101 **status** states the current state of the Inferior for the atomic business transaction,
2102 which corresponds to the last message sent to the Superior by (or in the case of
2103 ENROL for) the Inferior

2104

status value	meaning/previous message sent
<i>active</i>	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.
<i>inaccessible</i>	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

2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145

reply requested “true” if INFERIOR_STATE is sent as a query at the Superior’s initiative; “false” if INFERIOR_STATE is sent in reply to a received SUPERIOR_STATE or other message. Can only be “true” if “status” is “active” or “prepared-received”. Can only be “true” if “status” is “active”.

qualifiers standardised or other qualifiers.

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

A status of “unknown” shall only be sent if it has been determined for certain that the Inferior has no knowledge of a relationship with the Superior. If there could be persistent information corresponding to the Superior, but it is not accessible from the entity receiving an SUPERIOR_STATE/*y (or other) message targetted on the Inferior or the entity cannot determine whether any such persistent information exists, the response shall be “inaccessible”.

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

A SUPERIOR_STATE/INFERIOR_STATE exchange that determines that one or both sides are in the active state does not require that the Inferior be cancelled (unlike some other two-phase commit protocols). The relationship between Superior and Inferior, and related application elements may be continued, with new application messages carrying the same CONTEXT. Similarly, if the Inferior is prepared but the Superior is active, there is no required impact on the progression of the relationship between them.

The form INFERIOR_STATE/abcd refers to a INFERIOR_STATE message status having a value equivalent to “abcd” (for active, unknown and inaccessible) and with “reply requested” = “false”. INFERIOR_STATE/abcd/y refers to a similar message, but with “reply requested” = “true”. The form INFERIOR_STATE/*y refers to a INFERIOR_STATE message with “reply requested” = “true” and any value for status.

CHECK_STATUS

Sent to an Inferior to ask it to reply with STATUS.

Parameter

target address

Type

BTP address

<u>reply address</u>	<u>BTP address</u>
<u>inferior identifier</u>	<u>Identifier</u>
<u>Qualifiers</u>	<u>List of qualifiers</u>

2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165

~~target address the address to which the CHECK STATUS message is sent. This will be the address as inferior on the ENROL message.~~

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

~~inferior identifier This will be the "inferior identifier" on the ENROL message.~~

~~qualifiers standardised or other qualifiers.~~

~~Types of FAULT possible (sent to reply address)~~

~~General~~

STATUS

~~Sent by an Inferior in reply to a CHECK STATUS, reporting the overall state of the transaction tree node represented by the Inferior.~~

<u>Parameter</u>	<u>Type</u>
<u>target address</u>	<u>BTP address</u>
<u>address-as-inferior</u>	<u>BTP address</u>
<u>inferior identifier</u>	<u>Identifier</u>
<u>Status</u>	<u>See below</u>
<u>Qualifiers</u>	<u>List of qualifiers</u>

2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177

~~target address the address to which the STATUS is sent. This will be the reply address on the CHECK STATUS message.~~

~~address-as-inferior This will be the address as inferior as on the ENROL message (with the inferior identifier, this determines who the message is from).~~

~~inferior identifier This will be the inferior identifier as on the ENROL message.~~

~~status states the current status of the transaction tree node represented by the sender.~~

~~status value Meaning from Inferior~~

<u>status value</u>	<u>Meaning from Inferior</u>
<u>Created</u>	<u>The Inferior exists (and is addressable) but it has not been enrolled with a Superior</u>
<u>Enrolling</u>	<u>ENROL has been sent, but ENROLLED is awaited</u>
<u>Active</u>	<u>The Inferior is enrolled</u>
<u>Resigning</u>	<u>RESIGN has been sent; RESIGNED is awaited</u>
<u>Resigned</u>	<u>RESIGNED has been received</u>
<u>Preparing</u>	<u>PREPARE has been received; PREPARED has not been sent</u>
<u>Prepared</u>	<u>PREPARED has been sent; no outcome has been received or autonomous decision made</u>
<u>Confirming</u>	<u>CONFIRM has been received; CONFIRMED/responso has not bee sent</u>
<u>Confirmed</u>	<u>CONFIRMED/response has been sent</u>
<u>Cancelling</u>	<u>CANCEL has been received or auto-cancel has been decided</u>
<u>Cancelled</u>	<u>CANCELLED has been sent</u>
<u>cancel-contradiction</u>	<u>Autonomous cancel decision was made, CONFIRM received; CONTRADICTION has not been received</u>
<u>confirm-contradiction</u>	<u>Autonomous confirm decision was made, CANCEL received; CONTRADICTION has not been received</u>
<u>Hazard</u>	<u>A hazard has been discovered; CONTRADICTION has not been received</u>
<u>Contradicted</u>	<u>CONTRADICTION has been received</u>
<u>Unknown</u>	<u>No state information for the identifier exists; no such inferior exists</u>

<u>status value</u>	<u>Meaning from Inferior</u>
<u>Inaccessible</u>	<u>There may be state information for this identifier but it cannot be reached/existence cannot be determined.</u>

2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188

qualifiers standardised or other qualifiers.

REDIRECT

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

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

2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207

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

superior identifier The superior identifier as on the CONTEXT message and used on an ENROL message. (present only if the REDIRECT is sent from the Inferior).

inferior identifier The inferior identifier as on the ENROL message

old address The previous address of the sender of REDIRECT. A match is considered to apply if any of the old addresses match one that is already known.

new address The (set of alternatives) new addresses to be used for messages sent to this entity.

qualifiers standardised or other qualifiers.

2208
2209
2210
2211
2212
2213
2214
2215
2216

If the actor whose address is changed is an Inferior, the new address value replaces the address-as-inferior as present in the ENROL.

If the actor whose address is changed is a Superior, the new address value replaces the Superior address as present in the CONTEXT message (or as present in any other mechanism used to establish the Superior:Inferior relationship).

2217
2218

Messages involved used in control in the Initiator:Factory relationships

2219
2220

BEGIN

2221
2222
2223
2224
2225

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

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

2226

2227
2228
2229
2230

target address the address of the entity to which the BEGIN is sent. How this address is acquired and the nature of the entity are outside the scope of this specification.

2231
2232

reply address the address to which the replying BEGUN and related CONTEXT message should be sent.

2233
2234
2235
2236

transaction type identifies whether a new Cohesion or new Atom is to be created; this value will be the “superior type” in the new CONTEXT

2237
2238
2239
2240
2241

qualifiers standardised or other qualifiers. The standard qualifier “Transaction timelimit” may be present on BEGIN, to set the timelimit for the new business transaction and will be copied to the new CONTEXT. The standard qualifier “Inferior name” may be present if there is a CONTEXT related to the BEGIN.

2242
2243
2244
2245
2246

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

2247

2248
2249
2250

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

2251
2252
2253
2254

The forms BEGIN/cohesion and BEGIN/atom refer to BEGIN with “transaction type” having the corresponding value.

2255
2256

Types of FAULT possible (sent to Reply address)

2257
2258

General

2259
2260

BEGUN

2261
2262
2263

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

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

2264
2265
2266
2267

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

2268
2269

address-as-decider for a top-level transaction (no CONTEXT related to the BEGIN), this is the address to which

2270
2271

~~REQUEST_PREPARE~~PREPARE_INFERIORS,

2272
2273

~~REQUEST_CONFIRM~~CONFIRM_TRANSACTION,

2274
2275

~~REQUEST_CANCEL~~TRANSACTION, CANCEL_INFERIORS-and
REQUEST_INFERIOR_STATUSES messages are to be sent; if a CONTEXT was related to the BEGIN this parameter is absent

2276
2277

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

2278
2279

2280
2281

2282 inferior handle Shall be absent if this is a top-level transaction and may or may
2283 not be present otherwise. (Presence or absence will be determined by the nature
2284 of the Superior identified in the CONTEXT related to the BEGIN). If present, the
2285 inferior handle will identify this new business transaction as in the inferiors-list
2286 parameters in messages between the Superior identified in the CONTEXT related
2287 to the BEGIN (acting as a Decider) and its Terminator. The value shall be
2288 different for each enrolled Inferior of that Superior.

2289
2290 address-as-inferior This parameter shall be absent if this is a top-level
2291 transaction and may be present, at implementation option otherwise. If present, it
2292 shall be the address-as-inferior used in the enrolment with the Superior identified
2293 by the CONTEXT related to the BEGIN. If this is a top-level transaction

2294
2295 qualifiers standardised or other qualifiers.

2296
2297 At implementation option, the “address-as-decider” and/or “address-as-inferior” and the
2298 “address-as-superior” in the related CONTEXT may be the same or may be different. There
2299 is no general requirement that they even use the same bindings. Any may also be the same as
2300 the target address of the BEGIN message (the inferior identifier on messages will ensure they
2301 are applied to the appropriate Composer or Coordinator).

2302
2303 No FAULT messages are issued on receiving BEGUN.

2304 REQUEST_PREPAREPREPARE INFERIORS

2305
2306 Sent from a Terminator to a Decider, but only if it is a Cohesion Composer, to tell it to
2307 prepare all or some of its inferiors, by sending PREPARE to any that have not already sent
2308 PREPARED, RESIGN or CANCELLED to the Decider (Composer) on its relationships as
2309 Superior. If the inferiors-list parameter is absent, the request applies to all the inferiors; if the
2310 parameter is present, it applies only to the identified inferiors of the Decider (Composer).

2311

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

2312

2313 target address the address to which the
2314 REQUEST_PREPAREPREPARE INFERIORS message is sent. When sent
2315 from Terminator to Decider, †This will be the decider-address from the BEGUN
2316 message.

2317

2318 reply address When sent from Terminator to Decider, the address of the
2319 Terminator sending the PREPAREPREPARE INFERIORS message.

2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362

transaction identifier ~~When sent from Terminator to Decider,~~ identifies the Decider and will be the transaction-identifier from the BEGUN message.

inferiors-list ~~When sent from Terminator to Decider,~~ defines which of the Inferiors of this Decider preparation is requested for. If this parameter is absent ~~when sent to a Decider,~~ the PREPARE applies to all Inferiors.

qualifiers standardised or other qualifiers.

~~When sent to a Decider, f~~For all Inferiors identified in the inferiors-list parameter (all Inferiors if the parameter is absent), from which none of PREPARED, CANCELLED or RESIGNED has been received, the Decider shall issue PREPARE. It will reply to the Terminator, using the reply address on the ~~REQUEST_PREPARE~~PREPARE INFERIORS message, sending an INFERIOR_STATUSES message giving the status of the Inferiors identified on the inferiors-list parameter (all of them if the parameter was absent).

Types of FAULT possible (sent to Superior address)

General

InvalidDecider – if Decider address is unknown

UnknownTransaction – if the transaction-identifier is unknown

InvalidInferior – if an inferior-handle on the inferiors-list is unknown

WrongState – if a ~~REQUEST_CONFIRM~~CONFIRM TRANSACTION or CANCEL TRANSACTION/~~whole~~ has already been received by this Composer.

The form ~~REQUEST_PREPARE~~PREPARE INFERIORS/~~whole~~all refers to a ~~REQUEST_PREPARE~~PREPARE INFERIORS message sent to a Decider (Composer) where the “inferiors-list” parameter is absent. The form ~~REQUEST_PREPARE~~PREPARE INFERIORS/~~inferiors~~specific refers to a ~~REQUEST_PREPARE~~PREPARE INFERIORS message sent to a Decider (Composer) where the “inferiors-list” parameter is present.

REQUEST_CONFIRM_CONFIRM_TRANSACTION

Sent from a Terminator to a Decider to request confirmation of the business transaction. If the business transaction is a Cohesion, the confirm-set is specified by the “inferiors-list” parameter.

Parameter	Type
target address	BTP address

reply address	BTP address
transaction identifier	Identifier
inferiors-list	List of inferior handles
Report report-hazard	Boolean
Qualifiers	List of qualifiers

2363

2364

2365

2366

2367

target address the address to which the ~~REQUEST_CONFIRM~~CONFIRM_TRANSACTION message is sent. This will be the address-as-decider on the BEGUN message.

2368

2369

reply address the address of the Terminator sending the ~~REQUEST_CONFIRM~~CONFIRM_TRANSACTION message.

2370

2371

2372

transaction identifier identifies the Decider. This will be the transaction-identifier from the BEGUN message.

2373

2374

2375

2376

2377

inferiors-list defines which Inferiors enrolled with the Decider, if it is a Cohesion Composer, are to be confirmed. Shall be absent if the Decider is an Atom Coordinator.

2378

2379

2380

2381

2382

2383

2384

2385

report hazard Defines whether the Terminator wishes to be informed of hazard events and contradictory decisions within the business transaction. If “report hazard” is “true”, the receiver will wait until responses (CONFIRMED, CANCELLED or HAZARD) have been received from all of its inferiors, ensuring that any hazard events are reported. If “report hazard” is “false”, the Decider will reply with ~~CONFIRMED~~COMPLETE or ~~CANCELLED~~COMPLETE as soon as the decision for the transaction is known.

2386

2387

qualifiers standardised or other qualifiers.

2388

2389

2390

2391

2392

If the “inferiors-list” parameter is present, the Inferiors identified shall be the “confirm-set” of the Cohesion. If the parameter is absent and the business transaction is a Cohesion, the “confirm-set” shall be all remaining Inferiors. If the business transaction is an Atom, the “confirm-set” is automatically all the Inferiors.

2393

2394

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

2395

2396

If, for each of the Inferiors in the confirm-set, PREPARE has not been sent and PREPARED has not been received, PREPARE shall be issued to that Inferior.

2397

2398

2399

2400

NOTE -- If PREPARE has been sent but PREPARED not yet received from an Inferior in the confirm-set, it is an implementation option whether and when to re-send PREPARE. The Superior implementation may choose to re-

2401 send PREPARE if there are indications that the earlier PREPARE was not
2402 delivered.

2403
2404

2405 A confirm decision may be made only if PREPARED has been received from all Inferiors in
2406 the “confirm-set”. The making of the decision shall be persistent (and if it is not possible to
2407 persist the decision, it is not made). If there is only one remaining Inferior in the “confirm
2408 set” and PREPARE has not been sent to it, CONFIRM_ONE_PHASE may be sent to it.

2409
2410

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

2411
2412

If a confirm decision is made and “report-hazard” was “false”, a CONFIRMED_COMPLETE
2413 message shall be sent to the “reply-address”.

2414
2415

If a cancel decision is made and “report-hazard” was “false”, a CANCELLED_COMPLETE
2416 message shall be sent to the “reply-address”.

2417
2418

2419
2420

If “report-hazard” was “true” and any HAZARD or contradictory message was received (i.e.
2419 CANCELLED from an Inferior in the confirm-set or CONFIRMED from an Inferior not in
2420 the confirm-set), an INFERIOR_STATUSES reporting the status for all Inferiors shall be sent
2421 to the “reply-address”.

2422
2423

Types of FAULT possible (sent to reply address)

2424
2425

General

2426
2427

InvalidDecider – if Decider address is unknown

2428
2429

UnknownTransaction – if the transaction-identifier is unknown

2430
2431

InvalidInferior – if an inferior handle in the inferiors-list is unknown
WrongState – if a REQUEST_CANCEL_TRANSACTION~~whole~~ has
2431 already been received .

2432
2433

The form REQUEST_CONFIRM_CONFIRM_TRANSACTION~~whole~~all refers to a
2433 CONFIRM_TRANSACTION~~REQUEST_CONFIRM~~ message where the “inferiors-list”
2434 parameter is absent. The form CONFIRM_TRANSACTION~~REQUEST_CONFIRM~~
2435 /inferiors-specific refers to a CONFIRM_TRANSACTION~~REQUEST_CONFIRM~~ message
2436 where the “inferiors-list” parameter is present.

2437
2438

CONFIRM_COMPLETETRANSACTION_CONFIRMED

2439
2440

A Decider sends CONFIRM_COMPLETETRANSACTION_CONFIRMED to a Terminator
2441 in reply to REQUEST_CONFIRMCONFIRM_TRANSACTION if all of the confirm-set
2442 confirms (and, for a Cohesion, all other Inferiors cancel) without reporting hazards, or if the
2443 Decider made a confirm decision and the CONFIRM_TRANSACTION had a “report-
2444 hazards” value of “false”.

2445

Parameter

Type

<u>Parameter</u>	<u>Type</u>
<u>target address</u>	<u>BTP address</u>
<u>address-as-decider</u>	<u>BTP address</u>
<u>transaction-identifier</u>	<u>identifier</u>
<u>qualifiers</u>	<u>List of qualifiers</u>

2446

2447

2448

2449

2450

2451

2452

2453

2454

2455

2456

2457

2458

2459

2460

2461

2462

2463

2464

2465

2466

2467

2468

target address the address to which the ~~CONFIRM_COMPLETE~~TRANSACTION CONFIRMED is sent. ~~When sent from a Decider to a Terminator it~~this will be the reply address from the ~~REQUEST_CONFIRM~~CONFIRM TRANSACTION message.

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).

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).

qualifiers standardised or other qualifiers.

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

General

InvalidTerminator – if Terminator address is unknown

UnknownTransaction – if the transaction-identifier is unknown

2469

2470

2471

2472

~~Note – A CONFIRM_COMPLETE message arriving before a REQUEST_CONFIRM message is sent, or after a REQUEST_CANCEL has been sent will occur when the Decider has taken an autonomous decision and is not regarded as occurring in the wrong state.~~

2473

2474

2475

2476

2477

2478

CANCEL_TRANSACTION

Sent by a Terminator to a Decider at any time before CONFIRM_TRANSACTION has been sent.

<u>Parameter</u>	<u>Type</u>
<u>target address</u>	<u>BTP address</u>
<u>reply address</u>	<u>BTP address</u>

2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515

<u>transaction identifier</u>	<u>Identifier</u>
<u>report-hazard</u>	<u>Boolean</u>
<u>qualifiers</u>	<u>List of qualifiers</u>

target address the address to which the CANCEL TRANSACTION message is sent. This will be the decider-address from the BEGUN message.

reply address the address of the Terminator sending the CANCEL TRANSACTION message.

transaction identifier identifies the Decider and will be the transaction-identifier from the BEGUN message.

report hazard Defines whether the Terminator wishes to be informed of hazard events and contradictory decisions within the business transaction. If “report hazard” is “true”, the receiver will wait until responses (CONFIRMED, CANCELLED or HAZARD) have been received from all of its inferiors, ensuring that any hazard events are reported. If “report hazard” is “false”, the Decider will reply with TRANSACTION CANCELLED immediately.

qualifiers standardised or other qualifiers.

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.

Types of FAULT possible (sent to Superior address)

General

InvalidDecider – if Decider address is unknown

UnknownTransaction – if the transaction-identifier is unknown

WrongState – if a CONFIRM TRANSACTION has been received by this Composer.

REQUEST_CANCEL_CANCEL_INFERIORS

Sent by a Terminator to a Decider, but only if is a Cohesion Composer, at any time before REQUEST_CONFIRM CONFIRM TRANSACTION or CANCEL TRANSACTION has been sent.

<u>Parameter</u>	<u>Type</u>
<u>target address</u>	<u>BTP address</u>
<u>reply address</u>	<u>BTP address</u>

transaction identifier Identifier
inferiors-list List of inferior handles
qualifiers List of qualifiers

2516
2517 target address the address to which the
2518 ~~REQUEST_CANCEL~~CANCEL_TRANSACTION message is sent. ~~When sent~~
2519 ~~from Terminator to Decider,~~ This will be the decider-address from the BEGUN
2520 message.

2521
2522 reply address ~~When sent from Terminator to Decider,~~ the address of the
2523 Terminator sending the ~~REQUEST_CANCEL~~CANCEL_TRANSACTION
2524 message.

2525
2526 transaction identifier ~~When sent from Terminator to Decider,~~ identifies the
2527 Decider and will be the transaction-identifier from the BEGUN message.

2528
2529 inferiors-list ~~When sent from Terminator to Decider (Composer), defines which~~
2530 ~~of the Inferiors of this Decider are to be cancelled. This parameter shall be absent~~
2531 ~~when sent from a Terminator to a Decider (Coordinator).~~

2532
2533 qualifiers standardised or other qualifiers.

2534
2535 ~~When sent to a Decider with the inferiors-list parameter is absent, the business transaction is~~
2536 ~~cancelled—this is propagated to any remaining Inferiors by issuing CANCEL to them. No~~
2537 ~~more Inferiors will be permitted to enrol.~~

2538
2539 ~~When sent to a Decider (Composer), with the inferiors-list parameter present, only~~ Only the
2540 Inferiors identified in the inferiors-list are to be cancelled. Any other inferiors are unaffected
2541 by a ~~REQUEST_CANCEL~~CANCEL_INFERIORS/inferiors. Further Inferiors may be
2542 enrolled.

2543

2544 Note – A ~~REQUEST_CANCEL~~CANCEL_INFERIORS/inferiors issued to a
2545 ~~Decider (Cohesion Composer) identifying all of its~~the currently enrolled
2546 ~~Inferiors will leave the Cohesion 'empty', but permitted to continue with~~
2547 ~~new Inferiors, if any enrol.~~

2548
2549 Types of FAULT possible (sent to Superior address)

2550
2551 General
2552 InvalidDecider – if Decider address is unknown
2553 UnknownTransaction – if the transaction-identifier is unknown
2554 InvalidInferior – if an inferior-handle on the inferiors-list is unknown

2555 WrongState – if a REQUEST_CONFIRMCONFIRM TRANSACTION
2556 or CANCEL_TRANSACTION has been received by this Composer.

2557
2558 The form REQUEST_CANCEL/whole refers to a REQUEST_CANCEL message sent to a
2559 Decider where the “inferiors-list” parameter is absent. The form
2560 REQUEST_CANCEL/inferiors refers to a REQUEST_CANCEL message sent to a Decider
2561 (Composer) where the “inferiors-list” parameter is present.

2562
2563
2564 CANCEL_COMPLETETRANSACTION_CANCELLED

2565
2566 A Decider sends CANCEL_COMPLETETRANSACTION_CANCELLED to a Terminator in
2567 reply to REQUEST_CONFIRMCANCEL or in reply to CONFIRM_TRANSACTION if the
2568 Decider decided to cancel. In both cases, TRANSACTION_CANCELLED is used only -if all
2569 Inferiors canceled without reporting hazards or the CANCEL_TRANSACTION or
2570 CONFIRM_TRANSACTION had a “report-hazard” value of “false.”
2571

Parameter

<u>target address</u>	<u>BTP address</u>
<u>address-as-decider</u>	<u>BTP address</u>
<u>transaction-identifier</u>	<u>identifier</u>
<u>qualifiers</u>	<u>List of qualifiers</u>

2572
2573 target address the address to which the
2574 CANCEL_COMPLETETRANSACTION_CANCELLED is sent. When sent
2575 from a Decider to a Terminator it This will be the reply address from the
2576 CANCEL_TRANSACTION or
2577 REQUEST_CONFIRMCONFIRM_TRANSACTION message.

2578
2579 address-as-decider When the message is sent from a Decider to the
2580 Terminator, this shall be the address-as-decider of the Decider as on the BEGUN
2581 message (with the transaction identifier, this determines who the message is
2582 from).

2583
2584 transaction identifier When the message is sent from a Decider to the
2585 Terminator, this shall be the transaction identifier as on the BEGUN message (i.e.
2586 the identifier of the Decider as a whole).

2587
2588 qualifiers standardised or other qualifiers.

2589
2590 Types of FAULT possible (sent to address-as-decider)

2591
2592 General
2593 InvalidTerminator – if Terminator address is unknown

2594
2595

UnknownTransaction – if the transaction-identifier is unknown

2596
2597
2598
2599

Note – A CANCEL_COMPLETE message arriving before a REQUEST_CANCEL message is sent, or after a REQUEST_CONFIRM has been sent will occur when the Decider has taken an autonomous decision and is not regarded as occurring in the wrong state.

2600
2601

REQUEST_STATUSESREQUEST_INFERIOR_STATUSES

2602
2603
2604
2605
2606
2607
2608
2609
2610

Sent to a Decider to ask it to report the status of its Inferiors with an INFERIOR_STATUSES message. It can also be sent to any actor with an address-as-superior or address-as-inferior, asking it about the status of that transaction tree nodes Inferiors, if there are any. In this latter case, the receiver may reject the request with a FAULT(StatusRefused). If it is prepared to reply, but has no Inferiors, it replies with an INFERIOR_STATUSES with an empty “status-list” parameter.

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

2611

target address the address to which the REQUEST_STATUS message is sent. When used to a Decider, t~~t~~his will be the address-as-decider from the BEGUN message. Otherwise it may be an address-as-superior from a CONTEXT or address-as-inferior from an ENROL message.

2612
2613
2614
2615
2616

2617
2618

reply address the address to which the replying INFERIOR_STATUSES is to be sent

2619
2620

transaction-target identifier identifies the transaction (or transaction tree node) within the scope of the target address. Decider. When the message is used to a Decider, t~~t~~his will be the transaction-identifier from the BEGUN message. Otherwise it will be the superior-identifier from a CONTEXT or an inferior-identifier from an ENROL message.

2621
2622
2623
2624
2625

2626
2627

inferiors-list defines which inferiors enrolled with the Composer or Coordinator~~target~~ are to be included in the INFERIOR_STATUSES. If the list is absent, the status of all enrolled inferiors will be reported.

2628
2629

2630 **qualifiers** standardised or other qualifiers.

2631

2632 Types of FAULT possible (sent to reply-address)

2633

2634

General

2635

StatusRefused – if the receiver is not prepared to report its status to the sender of this message. This FAULT type shall not be issued when a Decider receives REQUEST_STATUSES from the Terminator.

2636

2637

InvalidDecider – if Decider address is unknown

2638

2639

UnknownTransaction – if the transaction-identifier is unknown

2640

2641

2642 The form REQUEST_STATUSESREQUEST INFERIOR STATUSES/all refers to a

2643

REQUEST_STATUS with the inferiors-list absent. The form

2644

REQUEST_INFERIOR_STATUS/specific inferiors refers to a

2645

REQUEST_INFERIOR_STATUS with the inferiors-list present.

2646

2647

INFERIOR_STATUSES

2648

2649

Sent by a Decider_ to report the status of all or some of its inferiors in response to a

2650

REQUEST_STATUSESREQUEST INFERIOR STATUSES,

2651

REQUEST_PREPAREPREPARE INFERIORS,

2652

REQUEST_CANCELCANCEL INFERIORS, CANCEL TRANSACTION with “report-

2653

hazard” value of “true”/inferiors and REQUEST_CONFIRMCONFIRM TRANSACTION

2654

with “report-hazard” value of “true”. It is also used by any actor in response to a received

2655

REQUEST_INFERIOR_STATUSES to report the status of inferiors, if there are any.

2656

Parameter	Type
target address	BTP address
<u>responders</u> address as decider	BTP address
<u>responders</u> transaction identifier	Identifier
status-list	Set of Status items - see below
general-qualifiers	List of qualifiers

2657

2658

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

2659

2660

2661

responders address ~~as decider~~ If the sender is a Decider, ~~the address-as-decider of the Decider~~ as on the BEGUN message. Otherwise the address of the sender of this message – one of address-as-inferior, address-as-superior. With the responders-identifier, this determines who the message is from. (with the transaction identifier, this determines who the message is from)

2662

2663

2664

2665

2666

2667
2668
2669
2670
2671
2672
2673
2674

transaction responders-identifier If the sender is a Decider, the transaction identifier as on the BEGUN message (i.e. the identifier of the Decider as a whole). Otherwise, the target-identifier used on the REQUEST_INFERIOR_STATUSES.

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).

2675
2676
2677
2678

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

status value	Meaning
<i>active</i>	The Inferior is enrolled
<i>resigned</i>	RESIGNED has been received from the Inferior
<i>preparing</i>	PREPARE has been sent to the inferior, none of PREPARED, RESIGNED, CANCELLED, HAZARD have been received
<i>prepared</i>	PREPARED has been received
<i>autonomously confirmed</i>	CONFIRMED/auto has been received, no completion message has been sent
<i>autonomously cancelled</i>	PREPARED had been received, and since then CANCELLED has been received but no completion message has been sent
<i>confirming</i>	CONFIRM has been sent, no outcome reply has been received
<i>confirmed</i>	CONFIRMED/response has been received
<i>cancelling</i>	CANCEL has been sent, no outcome reply has been received
<i>cancelled</i>	CANCELLED has been received, and PREPARED was not received previously

status value	Meaning
<i>cancel-contradiction</i>	Confirm had been ordered (and may have been sent), but CANCELLED was received
<i>confirm-contradiction</i>	Cancel had been ordered (and may have been sent) but CONFIRM/auto was received
<i>hazard</i>	A HAZARD message has been received
<u><i>invalid</i></u>	<u>No such inferior is enrolled (used only in reply to a REQUEST_INFERIOR_STATUSES/specific)</u>

2679

2680

2681

2682

2683

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.

2684

2685

2686

2687

2688

2689

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).

2690

2691

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

2692

2693

2694

2695

2696

2697

General

Invalid Terminator – if Terminator address is unknown

Unknown Transaction – if the transaction-identifier is unknown

2698

2699

REDIRECT

2700

2701

2702

2703

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

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

2704

2705 ~~target address~~ the address to which the REDIRECT is sent. This may be the
2706 ~~reply address from a received message or the address of the opposite side~~
2707 ~~(superior/inferior) as given in a CONTEXT or ENROL message~~
2708
2709 ~~superior identifier~~ The superior identifier as on the CONTEXT message and
2710 ~~used on an ENROL message. (present only if the REDIRECT is sent from the~~
2711 ~~Inferior).~~
2712
2713 ~~inferior identifier~~ The inferior identifier as on the ENROL message
2714
2715 ~~old address~~ The previous address of the sender of REDIRECT. A match is
2716 ~~considered to apply if any of the old addresses match one that is already known.~~
2717
2718 ~~new address~~ The (set of alternatives) new addresses to be used for messages
2719 ~~sent to this entity.~~
2720
2721 ~~qualifiers~~ standardised or other qualifiers.
2722
2723 ~~If the actor whose address is changed is an Inferior, the new address value~~
2724 ~~replaces the address as inferior as present in the ENROL.~~
2725
2726 ~~If the actor whose address is changed is a Superior, the new address value~~
2727 ~~replaces the Superior address as present in the CONTEXT message (or as present~~
2728 ~~in any other mechanism used to establish the Superior:Inferior relationship).~~
2729
2730
2731 **FAULT**
2732
2733 Sent in reply to various messages to report an error condition
2734

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

2735
2736 ~~target address~~ the address to which the FAULT is sent. This may be the reply
2737 ~~address from a received message or the address of the opposite side~~
2738 ~~(superior/inferior) as given in a CONTEXT or ENROL message~~
2739

2740
 2741
 2742
 2743
 2744
 2745
 2746
 2747
 2748
 2749
 2750
 2751

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

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

~~fault type~~ identifies the nature of the error, as specified for each of the main messages.

~~fault data~~ information relevant to the particular error. Each fault type defines the content of the fault data:

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 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 identifier.
DuplicateInferior	An inferior with the same address and identifier is already enrolled with this Superior.	The identifier.
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.

2752
 2753
 2754

~~qualifiers~~ standardised or other qualifiers.

2755
2756
2757

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

2758

Groups – combinations of related messages

2759
2760

2761
2762

The following combinations of messages form related groups, for which the meaning of the group is not just the aggregate of the meanings of the messages. The “&” notation is used to indicate relatedness. Messages appearing in parentheses in the names of groups in this section indicate messages that may or may not be present. The notation A & B / & C in a group name in this section indicates a group that contains A and B or A and C or A, B and C, possibly with any of those appearing more than once.

2763
2764

2765
2766

2767

CONTEXT & application message

2768
2769

2770

Meaning: the transmission of the application message is deemed to be part of the business transaction identified by the CONTEXT. The exact effect of this for application work implied by the transmission of the message is determined by the application – in many cases, it will mean the effects of the application message are to be subject to the outcome delivered to an enrolled Inferior, thus requiring the enrolment of a new Inferior if no appropriate Inferior is enrolled or if the CONTEXT is for cohesion.

2771
2772

2773

2774

2775

2776

2777

Target address: the target address is that of the application message. It is not required that the application address be a BTP address (in particular, there is no BTP-defined “additional information” field – the application protocol (and its binding) may or may not have a similar construct).

2778

2779

2780

2781

2782

There may be multiple application messages related to a single CONTEXT message. All the application messages so related are deemed to be part of the business transaction identified by the CONTEXT. This specification does not imply any further relatedness among the application messages themselves (though the application might).

2783

2784

2785

2786

The actor that sends the group shall retain knowledge of the Superior address in the CONTEXT. If the CONTEXT is a CONTEXT/atom, the actor shall also keep track of transmitted CONTEXTs for which no CONTEXT REPLY has been received.

2787

2788

2789

2790

If the CONTEXT is a CONTEXT/atom, the actor receiving the CONTEXT shall ensure that a CONTEXT REPLY message is sent back to the reply address of the CONTEXT with the appropriate completion status.

2791

2792

2793

2794

2795

Note – The representation of the relation between CONTEXT and one or more application messages depends on the binding to the carrier protocol. It is not necessary that the CONTEXT and application messages be closely associated “on the wire” (or even sent on the same connection) – some kind of referencing mechanism may be used.

2796

2797

2798

2799

2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846

CONTEXT_REPLY & ENROL

Meaning: the enrolment of the Inferior identified in the ENROL is to be performed with the Superior identified in the CONTEXT message this CONTEXT_REPLY is replying to. If the “completion-status” of CONTEXT_REPLY is “related”, failure of this enrolment shall prevent the confirmation of the business transaction.

Target address: the target address is that of the CONTEXT_REPLY. This will be the reply address of the CONTEXT message (in many cases, including request/reply application exchanges, this address will usually be implicit).

The target address of the ENROL message is omitted.

The actor receiving the related group will use the retained Superior address from the CONTEXT sent earlier to forward the ENROL. When doing so, it changes the ENROL to ask for a response (if it was an ENROL/no-rsp-req) and supplies its own address as the “reply-address”, remembering the original “reply-address” if there was one.

If ENROLLED is received and the original received ENROL was ENROL/rsp-req, the ENROLLED is forwarded back to the original “reply-address”.

If this attempt fails (i.e. ENROLLED is not received), and the “completion-status” of the CONTEXT_REPLY was “related”, the actor is required to ensure that the Superior does not proceed to confirmation. How this is achieved is an implementation option, but must take account of the possibility that direct communication with the Superior may fail. (One method is to prevent CONFIRM_TRANSACTION being sent to the Superior (in its role as Decider); another is to enrol as another Inferior before sending the original CONTEXT out with an application message). If the Superior is a sub-coordinator or sub-composer, an enrolment failure must ensure the sub-coordinator does not send PREPARED to its own Superior.

If the actor receiving the related group is also the Superior (i.e. it has the same binding address), the explicit forwarding of the ENROL is not required, but the resultant effect – that if enrolment fails the Superior does not confirm or issue PREPARED – shall be the same.

A CONTEXT_REPLY & ENROL group may contain multiple ENROL messages, for several Inferiors. Each ENROL shall be forwarded and an ENROLLED reply received before the Superior is allowed to confirm if the “completion-status” in the CONTEXT_REPLY was “related”.

When the group is constructed, if the CONTEXT had “superior-type” value of “atom”, the “completion-status” of the CONTEXT_REPLY shall be “related”. If the “superior-type” was “cohesive”, the “completion-status” shall be “completed” or “related” (as required by the application). If the value is “completed”, the actor receiving the group shall forward the ENROLs, but is not required to (though it may) prevent confirmation.

2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857

2858
2859

2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891

CONTEXT_REPLY (& ENROL) & PREPARED / & CANCELLED

This combination is characterised by a related CONTEXT_REPLY and either or both of PREPARED and CANCELLED, with or without ENROL.

Meaning: If ENROL is present, the meaning and required processing is the same as for CONTEXT_REPLY & ENROL. The PREPARED or CANCELLED message(s) are forwarded to the Superior identified in the CONTEXT message this CONTEXT_REPLY is replying to.

Note – the combination of CONTEXT_REPLY & ENROL & CANCELLED may be used to force cancellation of an atom

Target address: the target address is that of the CONTEXT_REPLY. This will be the reply address of the CONTEXT message (in many cases, including request/reply application exchanges, this address will usually be implicit).

The target address of the PREPARED and CANCELLED message is omitted – they will be sent to the Superior identified in the earlier CONTEXT message.

The actor receiving the group forwards the PREPARED or CANCELLED message to the Superior in as for an ENROL, using the retained Superior address from the CONTEXT sent earlier, except there is no reply required from the Superior.

If (as is usual) an ENROL and PREPARED or CANCELLED message are for the same Inferior, the ENROL shall be sent first, but the actor need not wait for the ENROLLED to come back before sending the PREPARED or CANCELLED (so an ENROL+PREPARED bundle from this actor to the Superior could be used).

The group can contain multiple ENROL, PREPARED and CANCELLED messages. Each PREPARED and CANCELLED message will be for a different Inferior.. There is no constraint on the order of their forwarding, except that ENROL and PREPARED or CANCELLED for the same Inferior shall be delivered to the Superior in the order ENROL first, followed by the other message for that Inferior.

CONTEXT_REPLY & ENROL & application message (& PREPARED)

This combination is characterised by a related CONTEXT_REPLY, ENROL and an application message. PREPARED may or may not be present in the related group.

Meaning: the relation between the BTP messages is as for the preceding groups. The transmission of the application message (and application effects implied by its

2892 transmission) has been associated with the Inferior identified by the ENROL and will be
2893 subject to the outcome delivered to that Inferior.

2894
2895 Target address: the target address of the group is the target address of the
2896 CONTEXT REPLY which shall also be the target address of the application message.
2897 The ENROL and PREPARED messages do not contain their target addresses.

2898
2899 The processing of ENROL and PREPARED messages is the same as for the previous
2900 groups.

2901
2902 This group can be used when participation in business transaction (normally a cohesion),
2903 is initiated by the service (Inferior) side, which fetches or acquires the CONTEXT, with
2904 some associated application semantic, performs some work for the transaction and sends
2905 an application message with a related ENROL. The CONTEXT REPLY allows the
2906 addressing of the application (and the CONTEXT REPLY) to be distinct from that of the
2907 Superior.

2908
2909 The actor receiving the group may associate the “inferior-handle” received on the
2910 ENROLLED with the application message in a manner that is visible to the application
2911 receiving the message.

2912

BEGUN & CONTEXT

2913

2914
2915 Meaning: the CONTEXT is that for the new business transaction, containing the
2916 Superior address.

2917

2918 Target address: the target address is that of the BEGUN message – this will be the reply
2919 address of the earlier BEGIN message.

2920

BEGIN & CONTEXT

2921

2922
2923 Meaning: the new business transaction is to be an Inferior (sub-coordinator or sub-
2924 composer) of the Superior identified by the CONTEXT. The Factory (receiver of the
2925 BEGIN) will perform the enrolment.

2926

2927 Target address: the target address is that of the BEGIN – this will be the address of the
2928 Factory.

2929

Standard qualifiers

2930

2931
2932 The following qualifiers are expected to be of general use to many applications and
2933 environments. The URI “urn:oasis:names:tc:BTP:qualifiers” is used in the
2934 Qualifier group value for the qualifiers defined here.

2935

2936

Transaction timelimit

2937

2938

2939 The transaction timelimit allows the Superior (or an application element initiating the
2940 business transaction) to indicate the expected length of the active phase, and thus give an
2941 indication to the Inferior of when it would be appropriate to initiate cancellation if the active
2942 phase appears to continue too long. The time limit ends (the clock stops) when the Inferior
2943 decides to be prepared and issues PREPARED to the Superior.
2944

2945 It should be noted that the expiry of the time limit does not change the permissible actions of
2946 the Inferior. At any time prior to deciding to be prepared (for an Inferior), the Inferior is
2947 **permitted** to initiate cancellation for internal reasons. The timelimit gives an indication to the
2948 entity of when it will be useful to exercise this right.
2949

2950 The qualifier is propagated on a CONTEXT message.
2951

2952 The “Qualifier name” shall be “transaction-timelimit” .
2953

2954 The “Content” shall contain the following field:
2955

Content field	Type
Timelimit	Integer

2956

2957 **Timelimit** indicates the maximum (further) duration, expressed as whole seconds from the
2958 time of transmission of the containing CONTEXT, of the active phase of the business
2959 transaction.
2960

2961 Inferior timeout

2962

2963 This qualifier allows an Inferior to limit the duration of its “promise”, when sending
2964 PREPARED, that it will maintain the ability to confirm or cancel the effects of all associated
2965 operations. Without this qualifier, an Inferior is expected to retain the ability to confirm or
2966 cancel indefinitely. If the timeout does expire, the Inferior is released from its promise and
2967 can apply the decision indicated in the qualifier.
2968

2969 It should be noted that BTP recognises the possibility that an Inferior may be forced to apply
2970 a confirm or cancel decision before the CONFIRM or CANCEL is received and before this
2971 timeout expires (or if this qualifier is not used). Such a decision is termed a heuristic decision,
2972 and (as with other transaction mechanisms), is considered to be an exceptional event. As with
2973 heuristic decisions, the taking of an autonomous decision by a Inferior **subsequent** to the
2974 expiry of this timeout, is liable to cause contradictory decisions across the business
2975 transaction. BTP ensures that at least the occurrence of such a contradiction will be
2976 (eventually) reported to the Superior of the business transaction. BTP treats “true” heuristic
2977 decisions and autonomous decisions after timeout the same way – in fact, the expiry in this
2978 timeout does not cause a qualitative (state table) change in what can happen, but rather a step
2979 change in the probability that it will.
2980

2981 The expiry of the timeout does not strictly require that the Inferior immediately invokes the
2982 intended decision, only that is at liberty to do so. An implementation may choose to only

2983 apply the decision if there is contention for the underlying resource, for example.
 2984 Nevertheless, Superiors are recommended to avoid relying on this and ensure decisions for
 2985 the business transaction are made before these timeouts expire (and allow a margin of error
 2986 for network latency etc.).

2987
 2988 The qualifier may be present on a PREPARED message. If the PREPARED message has the
 2989 “default is cancel” parameter “true”, then the “IntendedDecision” field of this qualifier shall
 2990 have the value “cancel”.

2991
 2992 The “Qualifier name” shall be “inferior-timeout” .

2993
 2994 The “Content” shall contain the following fields:

Content field	Type
Timeout	Integer
IntendedDecision	“confirm” or “cancel”

2996
 2997 **Timeout** indicates how long, expressed as whole seconds from the time of transmission of the
 2998 carrying message, the Inferior intends to maintain its ability to either confirm or cancel the
 2999 effects of the associated operations, as ordered by the receiving Superior.
 3000

3001 **IntendedDecision** indicates which outcome will be applied, if the timeout completes and an
 3002 autonomous decision is made.

3003
 3004 **Minimum inferior timeout**

3005
 3006 This qualifier allows a Superior to constrain the Inferior timeout qualifier received from the
 3007 Inferior. If a Superior knows that the decision for the business transaction will not be
 3008 determined for some period, it can require that Inferiors do not send PREPARED messages
 3009 with Inferior timeouts that would expire before then. An Inferior that is unable or unwilling to
 3010 send a PREPARED message with a longer (or no) timeout **should** cancel, and reply with
 3011 CANCELLED.
 3012

3013 The qualifier may be present on a CONTEXT, ENROLLED or PREPARE message. If
 3014 present on more than one, and with different values of the MinimumTimeout field, the value
 3015 on ENROLLED shall prevail over that on CONTEXT and the value on PREPARE shall
 3016 prevail over either of the others.
 3017

3018 The “Qualifier name” shall be “minimum-inferior-timeout” .

3019
 3020 The “Content” shall contain the following field:

Content field	Type
MinimumTimeout	Integer

3022

3023 **Minimum Timeout** is the minimum value of timeout, expressed as whole seconds, that will be
3024 acceptable in the Inferior timeout qualifier on an answering PREPARED message.

3025

3026 **Inferior name**

3027

3028 This qualifier allows an Enroller to supply a name for the Inferior that will be visible on
3029 INFERIOR_STATUSES and thus allow the Terminator to determine which Inferior (of the
3030 Composer or Coordinator) is related to which application work. This is in addition to the
3031 “inferior handle” field. The name can be human-readable and can also be used in fault
3032 tracing, debugging and auditing.

3033

3034 The name is never used by the BTP actors themselves to identify each other or to direct
3035 messages. (The BTP actors use the addresses and the identifiers in the message parameters
3036 for those purposes.)

3037

3038 This specification makes no requirement that the names are unambiguous within any scope
3039 (unlike the “inferior-handle” on ENROLLED and BEGUN, which is required to be
3040 unambiguous within the scope of the Decider). Other specifications, including those defining
3041 use of BTP with a particular application may place requirements on the use and form of the
3042 names. (This may include reference to information passed in application messages or in other,
3043 non-standardised, qualifiers.)

3044

3045 The qualifier may be present on BEGIN, ENROL and in the “qualifiers” field of a Status-item
3046 in INFERIOR_STATUSES. It is present on BEGIN only if there is a related CONTEXT; if
3047 present, the same qualifier value **should** be included in the consequent ENROL. If
3048 INFERIOR_STATUSES includes a Status-item for an Inferior whose ENROL had an
3049 inferior-name qualifier, the same qualifier value **should** be included in the Status-item.

3050

3051 The “Qualifier -name” shall be “inferior-name”

3052

3053 The “Content” shall contain the following fields:

3054

Content field	Type
inferior-name	String

3055

3056 **Inferior name** the name assigned to the enrolling Inferior.

3057

3058
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3099
3100
3101
3102
3103

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 to 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

3104 business transaction and on features of the implementation (e.g. making of a persistent record
3105 of the decision means that the information will survive at least some failures that otherwise
3106 lose state information, but the level of survival depends on the purpose of the
3107 implementation). [Table 2](#) and [Table 3](#) define the decision events.

3108
3109 In some cases, an implementation may not need to make an active change to have a persistent
3110 record of a decision, provided that the implementation will restore itself to the appropriate
3111 state on recovery. For example, an (inferior) implementation that “decided to be prepared”,
3112 and recorded a timeout (to cancel) in the persistent information for that decision (signalled via
3113 the appropriate qualifier on PREPARED), could treat the presence of an expired record as a
3114 record of “decide to cancel autonomously”, provided it always updated such a record as part
3115 of the “apply ordered confirmation” decision event.

3116
3117 The Superior event “decide to prepare” is considered semi-persistent. Since the sending of
3118 PREPARE indicates that the application exchange (to associate operations with the Inferior)
3119 is complete, it is not meaningful for the Superior:Inferior relationship to revert to an earlier
3120 state corresponding to an incomplete application exchange. However, implementations are
3121 not required to make the sending of PREPARE persistent in terms of recovery – a Superior
3122 that experiences failure after sending PREPARE may, on recovery, have no information
3123 about the transaction, in which case it is considered to be in the completed state (Z), which
3124 will imply the cancellation of the Inferior and its associated operations.

3125
3126 Where a Superior is itself an Inferior (to another Superior entity), in a hierarchic tree, its
3127 “decide to confirm” and “decide to cancel” decisions will in fact be the receipt of a
3128 CONFIRM or CANCEL instruction from its own Superior, without necessary change of local
3129 persistent information (which would combine both superior and inferior information, pointing
3130 both up and down the tree).

3131
3132

3133 Disruptions – failure events

3134

3135 Failure events are modelled as “disruption”. A failure and the subsequent recovery will (or
3136 may) cause a change of state. The disruption events in the state tables model different extents
3137 of loss of state information. An implementation is not required to exhibit all the possible
3138 disruption events, but it is not allowed to exhibit state transitions that do not correspond to a
3139 possible disruption.

3140

3141 In addition to the disruption events in the tables, there is an implicit “disruption 0” event,
3142 which involves possible interruption of service and loss of messages in transit, but no change
3143 of state (either because no state information was lost, or because recovery from persistent
3144 information restores the implementation to the same state). The “disruption 0” event would
3145 typically be an appropriate abstraction for a communication failure.

3146

3147 Invalid cells and assumptions of the communication mechanism

3148

3149 The empty cells in state table represent events that cannot happen. For events corresponding
3150 to sending a message or any of the decision events, this prohibition is absolute – e.g. a

3151 conformant implementation in the Superior active state “B1” will not send CONFIRM. For
3152 events corresponding to receiving a message, the interpretation depends on the properties of
3153 the underlying communications mechanism.

3154 For all communication mechanisms, it is assumed that

- 3156 a) the two directions of the Superior:Inferior communication are not synchronised –
3157 that is messages travelling in opposite directions can cross each other to any
3158 degree; any number of messages may be in transit in either direction; and
- 3159 b) messages may be lost arbitrarily

3160
3161 If the communication mechanisms guarantee ordered delivery (i.e. that messages, if delivered
3162 at all, are delivered to the receiver in the order they were sent) , then receipt of a message in a
3163 state where the corresponding cell is empty indicates that the far-side has sent a message out
3164 of order – a FAULT message with the Fault Type “WrongState” can be returned.

3165
3166 If the communication mechanisms cannot guarantee ordered delivery, then messages received
3167 where the corresponding cell is empty should be ignored. Assuming the far-side is
3168 conformant, these messages can assumed to be “stale” and have been overtaken by messages
3169 sent later but already delivered. (If the far-side is non-conformant, there is a problem
3170 anyway).

3171

3172 **Meaning of state table events**

3173

3174 The tables in this section define the events (rows) in the state tables. [Table 1](#) defines
3175 the events corresponding to sending or receiving BTP messages and the disruption events.
3176 [Table 2](#) describes the decision events for an Inferior, [Table 3](#) those for a
3177 Superior.

3178

3179 The decision events for a Superior, defined in [Table 3](#) cannot be specified without
3180 reference to other Inferiors to which it is Superior and to its relation with the application or
3181 other entity that (acting ultimately on behalf of the application) drives it.

3182

3183 The term “remaining Inferiors” refers to any actors to which this endpoint is Superior and
3184 which are to be treated as an atomic decision unit with (and thus including) the Inferior on
3185 this relationship. If the CONTEXT for this Superior:Inferior relationship had a “superior
3186 type” of “atom”, this will be all Inferiors established with same Superior address and Superior
3187 identifier except those from which RESIGN has been received. If the CONTEXT had
3188 “superior type” of “cohesion”, the “remaining Inferiors” excludes any that it has been
3189 determined will be cancelled, as well as any that have resigned – in other words it includes
3190 only those for which a confirm decision is still possible or has been made. The determination
3191 of exactly which Inferiors are “remaining Inferiors” in a cohesion is determined, in some
3192 way, by the application. The term “Other remaining Inferiors” excludes this Inferior on this
3193 relationship. A Superior with a single Inferior will have no “other remaining Inferiors”.

3194

3195 In order to ensure that the confirmation decision is delivered to all remaining Inferiors,
3196 despite failures, the Superior must persistently record which these Inferiors are (i.e. their
3197 addresses and identifiers). It must also either record that the decision is confirm, or ensure

3198 that the confirm decision (if there is one) is persistently recorded somewhere else, and that it
 3199 will be told about it. This latter would apply if the Superior were also BTP Inferior to another
 3200 entity which persisted a confirm decision (or recursively deferred it still higher). However,
 3201 since there is no requirement that the Superior be also a BTP Inferior to any other entity, the
 3202 behaviour of asking another entity to make (and persist) the confirm decision is termed
 3203 "offering confirmation" - the Superior offers the possible confirmation of itself, and its
 3204 remaining Inferiors to some other entity. If that entity (or something higher up) then does
 3205 make and persist a confirm decision, the Superior is "instructed to confirm" (which is
 3206 equivalent BTP CONFIRM).

3207
 3208 The application, or an entity acting indirectly on behalf of the application, may request a
 3209 Superior to prepare an Inferior (or all Inferiors). This typically implies that there will be no
 3210 more operations associated with the Inferior. Following a request to prepare all remaining
 3211 Inferiors, the Superior may offer confirmation to the entity that requested the prepare. (If the
 3212 Superior is also a BTP Inferior, its superior can be considered an entity acting on behalf of the
 3213 application.)

3214
 3215 The application, or an entity acting indirectly on behalf of the application, may also request
 3216 confirmation. This means the Superior is to attempt to make and persist a confirm decision
 3217 itself, rather than offer confirmation.
 3218
 3219

3220

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

Event name	Meaning
send/receive SUP_STATE/***/y	send/receive SUPERIOR_STATE with status *** and reply-requested = true ("prepared-rcvd" represents "prepared-received")
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

3221

3222

Table 2 : Decision events for Inferior

Event name	Meaning
decide to resign	<ul style="list-style-type: none"> Any associated operations have had no effect (data state is unchanged)).
decide to be prepared	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> As "decide to be prepared"; the persistent information specifies that the default action will be to cancel
decide to confirm autonomously	<ul style="list-style-type: none"> Decision to confirm autonomously has been made persistent; the effects of associated operations will be confirmed regardless of failures
decide to cancel autonomously	<ul style="list-style-type: none"> Decision to cancel autonomously has been made persistent the effects of associated operations will be cancelled regardless of failures
apply ordered confirmation	<ul style="list-style-type: none"> Effects of all associated operations have been confirmed; Persistent information is effectively removed
remove persistent information	<ul style="list-style-type: none"> Persistent information is effectively removed;

Event name	Meaning
detect problem	<ul style="list-style-type: none"> • For at least some of the associated operations, EITHER <ul style="list-style-type: none"> ○ they cannot be consistently cancelled or consistently confirmed; OR ○ it cannot be determined whether they will be cancelled or confirmed • AND, information about this is not persistent
detect and record problem	<ul style="list-style-type: none"> • 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)

3223

3224

Table 3: Decision events for a Superior

Event name	Meaning
decide to request confirm <u>one-phase</u>	<ul style="list-style-type: none"> • All associated application messages to be sent to the service have been sent; • There are no other remaining Inferiors • <u>If an atom, a</u>All enrolments that would create other Inferiors have completed (no outstanding CONTEXT_REPLYS) • The Superior has been requested to confirm
decide to prepare	<ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • Either <ul style="list-style-type: none"> ○ PREPARED or PREPARED/cancel has been received from all other remaining Inferiors; AND ○ Superior has been requested to confirm; AND ○ persistent information records the confirm decision and identifies all remaining Inferiors; • Or <ul style="list-style-type: none"> ○ persistent information records an offer of confirmation and has been instructed to confirm
decide to cancel	<ul style="list-style-type: none"> • Superior has not offered confirmation; OR • Superior has offered confirmation and has been instructed to cancel; OR

Event name	Meaning
	<ul style="list-style-type: none"> Superior has offered confirmation but has made an autonomous cancellation decision
remove confirm information	<ul style="list-style-type: none"> Persistent information has been effectively removed;
record contradiction	<ul style="list-style-type: none"> Information recording the contradiction has been persisted (to the degree considered appropriate)

3225

3226

3227

Persistent information

3228

3229

3230

3231

3232

3233

3234

Persisted information (especially prepared information at an Inferior, confirm information at a Superior) may include qualifications of the state carried in Qualifiers of the corresponding message (e.g. inferior timeouts in prepared information). It may also include application-specific information (especially in Inferiors) to allow the future confirmation or cancellation of the associated operations. In some cases it will also include information allowing an application message sent with a BTP message (e.g. PREPARED) to be repeated.

3235

3236

3237

3238

3239

The “effective” removal of persistent information allows for the possibility that the information is retained (perhaps for audit and tracing purposes) but some change to the persistent information (as a whole) means that if there is a failure after such change, on recovery, the persistent information does not cause the endpoint to return the state it would have recovered to before the change.

3240

3241

3242

3243

3244

3245

3246

In all cases, the degree to which information described as “persistent” will survive failure is a configuration and implementation option. An implementation **should** describe the level of failure that it is capable of surviving. For applications manipulating information that is itself volatile (e.g. network configurations), there is no requirement to make the BTP state information more persistent than the application information.

3247

3248

3249

3250

3251

3252

3253

The degree of persistence of the recording of a hazard (problem) at an Inferior and recording of a detected contradiction at a Superior may be different from that applying to the persistent prepared and confirm information. Implementations and configuration may choose to pass hazard and contradiction information via management mechanisms rather than through BTP. Such passing of information to a management mechanism could be treated as “record problem” or “record contradiction”.

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 <u>one-phase confirm</u> decided
Y1	completed queried
Z	completed and unknown

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 <u>ONE PHASE</u> received after prepared state
s2	REQUEST CONFIRM <u>ONE PHASE</u> received
s3	REQUEST CONFIRM <u>ONE PHASE</u> received, confirming
s4	REQUEST CONFIRM <u>ONE PHASE</u> received, cancelling
s5	REQUEST CONFIRM <u>ONE PHASE</u> received, hazard detected
s6	REQUEST CONFIRM <u>ONE PHASE</u> received, hazard recorded
x1	completed, presuming abort
x2	completed, presuming abort after prepared/cancel

State	summary
y1	completed, queried
y2	completed, default cancel, a message received
z	completed
z1	completed with default cancel

3257

3258

Table 6: Superior state table – normal forward progression

	I 1	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								F1	
send CONFIRM									
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_one-phase			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									

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_one-phase								
decide to prepare					F1	K1		
decide to confirm					L1	G4		
decide to cancel								
remove persistent information							R1	R1
record contradiction								
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		

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_one-phase								
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								

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_one-phase		
decide to prepare		
decide to confirm		
decide to cancel		
remove persistent information		
record contradiction		
disruption I	Z	
disruption II		
disruption III		
disruption IV		

3262

3263

3263

3264

Table 10: Inferior state table – normal forward progression

	i 1	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									

3265

3266

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 send CONFIRMED/response send CANCELLED send HAZARD			h1		j1		k1		l1	
send INF_STATE/active/y send INF_STATE/active send INF_STATE/unknown										
receive ENROLLED receive RESIGNED receive PREPARE receive CONFIRM_ONE_PHASE receive CONFIRM receive CANCEL receive CONTRADICTION receive SUP_STATE/active/y receive SUP_STATE/active receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd receive SUP_STATE/unknown			h1 s3 h2	h2	j1 s4 k1	j2	k1 k2	k2	l1 l2	l2
	g1	g2	l1		j2	j2				
			l2		k2		k2	k2	l2	l2
			h1		j1					
			h1		j1					
			h1		j1					
			h1		j1					
	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 detect problem detect and record problem				m1		z		z		z
	n1	n1								
	p2	p2								
disruption I disruption II disruption III	e1	e2	h1		j1		j1	k1 j1	h1	l1 h1

3267
3268

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			

3269

3270

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 send CANCELLED send HAZARD						
send INF_STATE/active/y send INF_STATE/active send INF_STATE/unknown						
receive ENROLLED receive RESIGNED receive PREPARE receive CONFIRM_ONE_PHASE receive CONFIRM receive CANCEL receive CONTRADICTION receive SUP_STATE/active/y receive SUP_STATE/active receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd receive SUP_STATE/unknown	s1	s2	s3	s4	s5	s6
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			s3			s6
disruption I disruption II disruption III	e1	z		z	z	

3271

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 send HAZARD						z1
send INF_STATE/active/y send INF_STATE/active send INF_STATE/unknown					z	
receive ENROLLED receive RESIGNED receive PREPARE receive CONFIRM_ONE_PHASE receive CONFIRM receive CANCEL receive CONTRADICTION receive SUP_STATE/active/y receive SUP_STATE/active receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd receive SUP_STATE/unknown						z z y1 z1 y1 y1 m1 y2 y1 y1 z z y1 y2 z z1 y2 y2 y2 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 detect problem detect and record problem						z z
disruption I disruption II disruption III	e1	e2				

3272

3273

3274

3274 **Failure Recovery**

3275 **Types of failure**

3276

3277 BTP is designed to ensure the delivery of a consistent decision for a business transaction to
3278 the parties involved, even in the event of failure. Failures can be classified as:

3279

3280 **Communication failure:** messages between BTP actors are lost and not
3281 delivered. BTP assumes the carrier protocol ensures that messages are either
3282 delivered correctly (without corruption) or are lost, but does not assume that all
3283 losses are reported or that messages sent separately are delivered in the order of
3284 sending.

3285

3286 **Node failure (system failure, site failure):** a machine hosting one or more BTP
3287 actors stops processing and all its volatile data is lost. BTP assumes a site fails by
3288 stopping – it either operates correctly or not at all, it never operates incorrectly.

3289

3290 Communication failure may become known to a BTP implementation by an indication from
3291 the lower layers or may be inferred (or suspected) by the expiry of a timeout. Recovery from
3292 a communication failure requires only that the two actors can again send messages to each
3293 other and continue or complete the progress of the business transaction. In the state tables for
3294 the Superior:Inferior relationship, each side is either waiting to make a decision or can send a
3295 message. For some states, the message to be sent is a repetition of a regular message; for
3296 other states, the INFERIOR_STATE or SUPERIOR_STATE message can be sent, requesting
3297 a response. Thus, following a communication failure, either side can prompt the other to re-
3298 establish the relationship. Receiving one of the *_STATE messages asking for a response
3299 does not require an immediate response – especially if an implementation is waiting to
3300 determine a decision (perhaps because it is itself waiting for a decision from elsewhere), an
3301 implementation may choose not to reply until it wishes too.

3302

3303 A node failure is distinguished from communication failure because there is loss of volatile
3304 state. To ensure consistent application of the decision of a business transaction, BTP requires
3305 that some state information will be persisted despite node failure. Exactly what real events
3306 correspond to node failure but leave the persistent information undamaged is a matter for
3307 implementation choice, depending on application requirements; however, for most
3308 application uses, power failure should be survivable (an exception would be if the data
3309 manipulated by the associated operations was volatile). There will always be some level of
3310 event sufficiently catastrophic to lose persistent information and the ability to recover–
3311 destruction of the computer or bankruptcy of the organisation, for example.

3312

3313 Recovery from node failure involves recreating the endpoint in a node that has access to the
3314 persistent information for incomplete transactions. This may be a recreation of the original
3315 node (including the ability to perform application work) using the same addresses; or there
3316 may be a distinct recovery entity, which can access the persistent data, but has a different
3317 address; other implementation approaches are possible. Restoration of the endpoint from
3318 persistent information will often result in a partial loss of state, relative to the volatile state
3319 reached before the failure. This is modelled in the state tables by the “disruption” events.

3320 After recovery from node failure, the implementation behaves much as if a communication
3321 failure had occurred.

3322

3323 Persistent information

3324

3325 BTP requires that some decision events are persisted – that information recording an
3326 Inferior’s decision to be prepared, a Superior’s decision to confirm and an Inferior’s
3327 autonomous decision survive failure. Making the first two decisions persistent ensures that a
3328 consistent decision can be reached for the business transaction and that it is delivered to all
3329 involved nodes. Requiring an Inferior’s autonomous decision to be persistent allows BTP to
3330 ensure that, if this decision is contradictory (i.e. opposite to the decision at the Superior), the
3331 contradiction will be reported to the Superior, despite failures.

3332

3333 BTP also permits, but does not require, recovery of the Superior:Inferior relationship in the
3334 active state (unlike many transaction protocols, where a communication or endpoint failure in
3335 active state would invariably cause rollback of the transaction). Recovery in the active state
3336 may require that the application exchange is resynchronised as well – BTP does not directly
3337 support this, but does allow continuation of the business transaction as such. In the state
3338 tables, from some states, there are several levels of disruption, distinguished by which state
3339 the implementation transits to – this represents the survival of different extents of state
3340 information over failure and recovery. The different levels of disruption describe legitimate
3341 states for the endpoint to be in after it has recovered – **they do not require that all**
3342 **implementations are able to exhibit the appropriate partial loss of state information.**

3343 The absence of a destination state for the disruption events means that such a transition is not
3344 legitimate – thus, for example, an Inferior that has decided to be prepared will always recover
3345 to the same state, by virtue of the information persisted in the “decide to be prepared” event.

3346

3347 Apart from the (optional) recovery in active state, BTP follows the well-known presume-
3348 abort model – it is only required that information be persisted when decisions are made (and
3349 not, e.g. on enrolment). This means that on recovery, one side may have persistent
3350 information but the other does not. This occurs when an Inferior has decided to be prepared
3351 but the Superior never confirmed (so the decision is “presumed” to be cancel), or because the
3352 Superior did confirm, and the Inferior applied the confirm, removed its persistent information
3353 but the acknowledgement (CONFIRMED) was never received by the Superior (or, at least, it
3354 still had the persistent information when the failure occurred).

3355

3356 Information to be persisted for an Inferior’s “decision to be prepared” must be sufficient to
3357 re-establish communication with the Superior, to apply a confirm decision and to apply a
3358 cancel decision. It will thus need to include

3359 Inferior identity (this may be an index used to locate the information)

3360 Superior address (as on CONTEXT)

3361 Superior identifier (as on CONTEXT)

3362 default-is-cancel value (as on PREPARED)

3363

3364 The information needed to apply confirm/cancel decisions will depend on the application and
3365 the associated operations. It may also normally be necessary to persist any qualifiers that

3366 were sent with the PREPARED message or application messages sent with the PREPARED,
3367 since the PREPARED message will be repeated if a failure occurs.

3368
3369 A Superior must record corresponding information to allow it to re-establish communication
3370 with the Inferior:

3371 Inferior address (as on ENROL)

3372 Inferior identifier (as on ENROL)

3373

3374 A Superior that is the Decider for the business transaction need only persist this information
3375 if it makes a decision to confirm (and this Inferior is in the confirm set, for a Cohesion). A
3376 Superior that is also an Inferior to some other entity (i.e. it is an intermediate in a tree, as
3377 atom in a cohesion, sub-coordinator or sub-composer) must persist this information as
3378 Superior (to this Inferior) as part of the persistent information of its decision to be prepared
3379 (as an Inferior). For such an entity, the “decision to confirm” as Superior is made when (and
3380 if) CONFIRM is received from its Superior or it makes an autonomous decision to confirm. If
3381 CONFIRM is received, the persistent information may be changed to show the confirm
3382 decision, but alternatively, the receipt of the CONFIRM can be treated as the decision itself.
3383 If the persistent information is left unchanged and there is a node failure, on recovery the
3384 entity (as an Inferior) will be in a prepared state, and will rediscover the confirm decision
3385 (using the recovery exchanges to its Superior) before propagating it to its Inferior(s).

3386

3387 After failure, an implementation may not be able to restore an endpoint to the appropriate
3388 state immediately – in particular, the necessary persistent information may be inaccessible,
3389 although the implementation can respond to received BTP messages. In such a case, a
3390 Superior may reply to any BTP message except INFERIOR_STATE/* (i.e. with a “reply-
3391 requested” value “false”) with SUPERIOR_STATE/inaccessible and an Inferior to any BTP
3392 message except SUPERIOR_STATE/* with “INFERIOR_STATE/inaccessible. Receipt of
3393 the *_STATE/inaccessible messages has no effect on the endpoint state.

3394

3395 Redirection

3396

3397 As described above, BTP uses the presume-abort model for recovery. A corollary of this is
3398 that there are cases where one side will attempt to re-establish communication when there is
3399 no persistent information for the relationship at the far-end. In such cases, it is important the
3400 side that is attempting recovery can distinguish between unsuccessful attempts to connect to
3401 the holder of the persistent information and when the information no longer exists. If the peer
3402 information does not exist, this side can draw conclusions and complete appropriately; if they
3403 merely fail to get through they are stuck in attempting recovery.

3404

3405 Two mechanisms are provided to make it possible that even when one side of a
3406 Superior:Inferior relationship has completed, that a message can eventually get through to
3407 something that can definitively report the status, distinguishing this case from a temporary
3408 inability to access the state of a continuing transaction element. The mechanisms are:

- 3409 o Address fields which provide a “callback address” can be a set of addresses,
3410 which are alternatives one of which is chosen as the target address for the
3411 future message. If the sender of that message finds the address does not work,
3412 it can try a different alternative.

- 3413 o The REDIRECT message can be used to inform the peer that an address
3414 previously given is no longer valid and to supply a replacement address (or
3415 set of addresses). REDIRECT can be issued either as a response to receipt of
3416 a message or spontaneously.

3417
3418 The two mechanisms can be used in combination, with one or more of the original set of
3419 addresses just being a redirector, which does not itself ever have direct access to the state
3420 information for the transaction, but will respond to any message with an appropriate
3421 REDIRECT.

3422
3423 An alternative implementation approach is to have a single addressable entity that uses the
3424 same address for all transactions, distinguishing them by identifier, and which always
3425 recovers to use the same address. Such an implementation would not need to supply
3426 “backup” addresses (and would only use REDIRECT if it was being permanently migrated).
3427

3428 Terminator:Decider failures

3429
3430 BTP does not provide facilities or impose requirements on the recovery of
3431 Terminator:Decider relationships, other than allowing messages to be repeated. A Terminator
3432 may survive failures (by retaining knowledge of the Decider’s address and identifier), but this
3433 is an implementation option. Although a Decider (if it decides to confirm) will persist
3434 information about the confirm decision, it is not required, after failure, to remain accessible
3435 using the inferior address it offered to the Terminator. Any such recovery is an
3436 implementation option.

3437
3438 A Decider’s address (as returned on BEGUN) may be a set of addresses, allowing a failed
3439 Decider to be recovered at a different address.

3440
3441 A Decider has no way of initiating a call to a Terminator to ensure that it is still active, and
3442 thus no way of detecting that a Terminator has failed. To avoid a Decider waiting for ever for
3443 a ~~REQUEST_CONFIRM~~CONFIRM_TRANSACTION that will never arrive, the standard
3444 qualifier “Transaction timelimit” can be used (by the Initiator) to inform the Decider when it
3445 can assume the Terminator will not issue
3446 ~~REQUEST_CONFIRM~~CONFIRM_TRANSACTION and so it (the Decider) should initiate
3447 cancellation.
3448

3449 XML representation of Message Set

3450
3451 This section describes the syntax for BTP messages in XML. These XML messages represent
3452 a midpoint between the abstract messages and what actually gets sent on the wire.

3453
3454 All BTP related URIs have been created using Oasis URI conventions as specified in [RFC](#)
3455 [3121](#)

3456
3457 The XML Namespace for the BTP messages is urn:oasis:names:tc:BTP:xml
3458

3459 In addition to an XML schema, this specification uses an informal syntax to describe the
3460 structure of the BTP messages. The syntax appears as an XML instance, but the values
3461 contain data types instead of values. The following symbols are appended to some of the
3462 XML constructs: ? (zero or one), * (zero or more), + (one or more.) The absence of one of
3463 these symbols corresponds to "one and only one."
3464

3465 Addresses

3466
3467 As described in the "Abstract Message and Associated Contracts – Addresses" section, a BTP
3468 address comprises three parts, and for a target address only the "additional information" field
3469 is inside the BTP messages. For all BTP messages whose abstract form includes a target
3470 address parameter, the corresponding XML representation includes a "target-additional-
3471 information" element. This element may be omitted if it would be empty.
3472

3473 For other addresses, all three fields are present, as in:

```
3474 <btp:some-address>  
3475   <btp:binding-name>...carrier binding URI...</btp:binding-name>  
3476   <btp:binding-address>...carrier specific  
3477   address...</btp:binding-address>  
3478   <btp:additional-information>...optional additional addressing  
3479   information...</btp:additional-information> ?  
3480 </btp:some-address>
```

3481
3482
3483
3484 A "published" address can be a set of <some-address>, which are alternatives which can be
3485 chosen by the peer (sender.) Multiple addresses are used in two cases: different bindings to
3486 same endpoint, or backup endpoints. In the former, the receiver of the message has the choice
3487 of which address to use (depending on which binding is preferable.) In the case where
3488 multiple addresses are used for redundancy, a priority attribute can be specified to help the
3489 receiver choose among the addresses- the address with the highest priority should be used,
3490 other things being equal. The priority is used as a hint and does not enforce any behaviour in
3491 the receiver of the message. Default priority is a value of 1.
3492

3493 Qualifiers

3494 The "Qualifier name" is used as the element name, within the namespace of the "Qualifier
3495 group".
3496

3497 Examples:

```
3498 <btpq:inferior-timeout  
3499   xmlns:btpq="urn:oasis:names:tc:BTP:qualifiers"  
3500   xmlns:btp="urn:oasis:names:tc:BTP:xml "  
3501   btp:must-be-understood="false "  
3502   btp:to-be-  
3503   propagated="false">1800</auth:usernamebtpq:inferior-timeout> |  
3504  
3505 <auth:username  
3506   xmlns:auth="http://www.example.com/ns/auth"  
3507   xmlns:btp="urn:oasis:names:tc:BTP:xml "
```

3508
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
3530
3531
3532
3533
3534
3535
3536
3537
3538
3539
3540
3541
3542
3543
3544
3545
3546
3547
3548
3549
3550
3551
3552
3553
3554
3555
3556

```
btp:must-be-understood="true"  
btp:to-be-propagated="true">jtauber</auth:username>
```

Attributes must-be-understood **has default value “true”** and to-be-propagated has default value “false”.

Identifiers

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

Examples: "01", "FAB224234CCCC2"

Note – Use of hexadecimal digits avoids problems with character-code representations. The only operation the BTP implementations have to perform on identifiers is to match them.

Message References

Each BTP message has an optional id attribute to give it a unique identifier. An application can make use of those identifiers, but no processing is enforced.

Messages

CONTEXT

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

CONTEXT _REPLY

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

```
3557 <completion-status>completed|related|repudiated</completion-
3558 status>
3559 <btp:qualifiers> ?
3560 ...qualifiers...
3561 </btp:qualifiers>
3562 </btp:context>
```

BEGIN

```
3565
3566
3567 <btp:begin id? transaction-type="cohesion|atom">
3568 <btp:target-additional-information>
3569 ...additional address information...
3570 </btp:target-additional-information>
3571 <btp:reply-address>
3572 ...address...
3573 </btp:reply-address>
3574 <btp:qualifiers> ?
3575 ...qualifiers...
3576 </btp:qualifiers>
3577 </btp:begin>
```

BEGUN

```
3580
3581
3582 <btp:begun id? transaction-type="cohesion|atom">
3583 <btp:target-additional-information>
3584 ...additional address information...
3585 </btp:target-additional-information>
3586 <btp:decider-address> ?
3587 ...address...
3588 </btp:decider-address>
3589 <btp:transaction-identifier>...hexstring...</btp:transaction-
3590 identifier> ?
3591 <btp:inferior-handle>...hexstring...</btp:inferior:handle> ?
3592 <btp:inferior-address> ?
3593 ...address...
3594 </btp:inferior-address>
3595 <btp:qualifiers> ?
3596 ...qualifiers...
3597 </btp:qualifiers>
3598 </btp:begun>
```

ENROL

```
3601
3602
3603 <btp:enrol reply-requested="true|false" id?>
3604 <btp:target-additional-information>
3605 ...additional address information...
3606 </btp:target-additional-information>
```

```

3607 <btp:superior-identifier>...hexstring...</btp:superior-
3608 identifier>
3609 <btp:reply-address> ?
3610 ...address...
3611 </btp:reply-address>
3612 <btp:inferior-address> +
3613 ...address...
3614 </btp:inferior-address>
3615 <btp:inferior-identifier>...hexstring...</btp:inferior-
3616 identifier>
3617 <btp:qualifiers> ?
3618 ...qualifiers...
3619 </btp:qualifiers>
3620 </btp:enrol>
3621
3622

```

ENROLLED

```

3623
3624
3625 <btp:enrolled id?>
3626 <btp:target-additional-information>
3627 ...additional address information...
3628 </btp:target-additional-information>
3629 <btp:inferior-identifier>...hexstring...</btp:inferior-
3630 identifier>
3631 <btp:inferior-handle>...hexstring...</btp:inferior:handle> ?
3632 <btp:qualifiers> ?
3633 ...qualifiers...
3634 </btp:qualifiers>
3635 </btp:enrolled>
3636
3637

```

RESIGN

```

3638
3639
3640 <btp:resign response-requested="true|false" id?>
3641 <btp:target-additional-information>
3642 ...additional address information...
3643 </btp:target-additional-information>
3644 <btp:superior-identifier>...hexstring...</btp:superior-
3645 identifier>
3646 <btp:inferior-address> +
3647 ...address...
3648 </btp:inferior-address>
3649 <btp:inferior-identifier>...hexstring...</btp:inferior-
3650 identifier>
3651 <btp:qualifiers> ?
3652 ...qualifiers...
3653 </btp:qualifiers>
3654 </btp:resign>
3655
3656

```

3657 **RESIGNED**

3658

3659

3660

3661

3662

3663

3664

3665

3666

3667

3668

3669

3670

```
<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>
```

3671 **PREPARE**

3672

3673

3674

3675

3676

3677

3678

3679

3680

3681

3682

3683

3684

3685

3686

3687

3688

3689

3690

3691

3692

3693

```
<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>
<btp:transaction-identifier>...hexstring...</btp:transaction-
identifier> ?
<btp:inferiors-list> ?
<btp:inferior-handle>...hexstring...</btp:inferior-handle>
+
</btp:inferiors-list>
  <btp:qualifiers> ?
    ...qualifiers...
  </btp:qualifiers>
</btp:prepare>
```

3694 **PREPARED**

3695

3696

3697

3698

3699

3700

3701

3702

3703

3704

3705

3706

3707

3708

```
<btp:prepared default-is-cancel="false|true" 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...
```

3709
3710
3711
3712

```
</btp:qualifiers>  
</btp:prepared>
```

CONFIRM

3713
3714

3715
3716
3717
3718
3719
3720

```
<btp:confirm 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:confirm>
```

3721
3722
3723
3724
3725
3726

CONFIRMED

3727
3728

3729
3730
3731
3732
3733
3734

```
<btp:confirmed confirmed-received="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:decider-address> ?  
  ..address...  
  </btp:decider-address>  
  <btp:transaction-identifier>...hexstring...</btp:transaction-  
  identifier> ?  
  <btp:qualifiers> ?  
    ..qualifiers...  
  </btp:qualifiers>  
</btp:confirmed>
```

3735
3736
3737
3738
3739

3740
3741
3742
3743
3744

3745
3746
3747
3748
3749
3750

CANCEL

3751
3752

3753
3754
3755
3756
3757
3758
3759

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

```

3760     ...address...
3761     </btp:reply-address>
3762     <btp:transaction-identifier>...hexstring...</btp:transaction-
3763 identifier> ?
3764     <btp:inferiors-list> ?
3765     <btp:inferior-handle>...hexstring...</btp:inferior-handle>
3766     </btp:inferiors-list>
3767     <btp:qualifiers> ?
3768     ..qualifiers...
3769     </btp:qualifiers>
3770 </btp:cancel>

```

CANCELLED

```

3773
3774
3775 <btp:cancelled id?>
3776   <btp:target-additional-information>
3777     ..additional address information...
3778   </btp:target-additional-information>
3779   <btp:superior-identifier>...hexstring...</btp:superior-
3780 identifier>
3781   <btp:inferior-address> +
3782     ..address...
3783   </btp:inferior-address> ?
3784   <btp:inferior-identifier>...hexstring...</btp:inferior-
3785 identifier> ?
3786   <btp:decider address> ?
3787   ..address...
3788   </btp:decider address>
3789   <btp:transaction identifier>...hexstring...</btp:transaction-
3790 identifier> ?
3791   <btp:qualifiers> ?
3792   ..qualifiers...
3793   </btp:qualifiers>
3794 </btp:cancelled>

```

CONFIRM_ONE_PHASE

```

3797
3798
3799 <btp:confirm-one-phase report-hazard="true|false" id?>
3800 <btp:target-additional-information>
3801 ..additional address information...
3802 </btp:target-additional-information>
3803 <btp:inferior-identifier>...hexstring...</btp:inferior-
3804 identifier>
3805 <btp:qualifiers> ?
3806 ..qualifiers...
3807 </btp:qualifiers>
3808 </btp:confirm-one-phase>
3809

```

HAZARD

3810
3811
3812
3813
3814
3815
3816
3817
3818
3819
3820
3821
3822
3823
3824
3825
3826
3827
3828

```
<btp:hazard level="mixed|possible" 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:hazard>
```

CONTRADICTION

3829
3830
3831
3832
3833
3834
3835
3836
3837
3838
3839
3840
3841
3842

```
<btp:contradiction 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:contradiction>
```

SUPERIOR_STATE

3843
3844
3845
3846
3847
3848
3849
3850
3851
3852
3853
3854
3855
3856
3857
3858

```
<btp:superior-state reply-requested="true|false" id?>
  <btp:target-additional-information>
    ...additional address information...
  </btp:target-additional-information>
  <btp:inferior-identifier>...hexstring...</btp:inferior-
identifier>
  <btp:status>active|prepared-
received|inaccessible|unknown</btp:status>
  <btp:qualifiers> ?
    ...qualifiers...
  </btp:qualifiers>
</btp:superior-state>
```

INFERIOR_STATE

3859
3860

```

3861 <btp:inferior-state reply-requested="true|false" id?>
3862   <btp:target-additional-information>
3863     ...additional address information...
3864   </btp:target-additional-information>
3865   <btp:superior-identifier>...hexstring...</btp:superior-
3866   identifier>
3867   <btp:inferior-address> +
3868     ...address...
3869   </btp:inferior-address>
3870   <btp:inferior-identifier>...hexstring...</btp:inferior-
3871   identifier>
3872   <btp:status> active|_prepared-
3873   received+inaccessible|unknown</btp:status>
3874   <btp:qualifiers> ?
3875     ...qualifiers...
3876   </btp:qualifiers>
3877 </btp:inferior-state>

```

CHECK STATUS

```

3880
3881
3882 <btp:check_status id?>
3883 <btp:target_additional_information>
3884 ...additional address information...
3885 </btp:target_additional_information>
3886 <btp:reply-address>
3887 ...address...
3888 </btp:reply address>
3889 <btp:inferior identifier>...hexstring...</btp:inferior
3890 identifier> ?
3891 <btp:qualifiers> ?
3892 ...qualifiers...
3893 </btp:qualifiers>
3894 </btp:check_status>

```

STATUS

```

3899 <btp:status id?>
3900 <btp:target_additional_information>
3901 ...additional address information...
3902 </btp:target_additional_information>
3903 <btp:inferior address> ?
3904 ...address...
3905 </btp:inferior address>
3906 <btp:inferior identifier>...hexstring...</btp:inferior
3907 identifier> ?
3908 <btp:status_value>created|enrolling|active|resigning|
3909 resigned|preparing|prepared|
3910 confirming|confirmed|cancelling|cancelled|
3911 cancel contradiction|confirm contradiction|

```

```
3912 hazard|contradicted|unknown|inaccessible</btp:status-  
3913 value>  
3914 <btp:qualifiers> ?  
3915 ...qualifiers...  
3916 </btp:qualifiers>  
3917 </btp:status>
```

REDIRECT

```
3920  
3921  
3922 <btp:redirect id?>  
3923 <btp:target-additional-information>  
3924 ...additional address information...  
3925 </btp:target-additional-information>  
3926 <btp:superior-identifier>...hexstring...</btp:superior-  
3927 identifier> ?  
3928 <btp:inferior-identifier>...hexstring...</btp:inferior-  
3929 identifier>  
3930 <btp:old-address> +  
3931 ...address...  
3932 </btp:old-address>  
3933 <btp:new-address> +  
3934 ...address...  
3935 </btp:new-address>  
3936 <btp:qualifiers> ?  
3937 ...qualifiers...  
3938 </btp:qualifiers>  
3939 </btp:redirect>
```

CONFIRM_ONE_PHASE

```
3942  
3943  
3944 <btp:confirm one phase report hazard="true|false" id?>  
3945 <btp:target-additional-information>  
3946 ...additional address information...  
3947 </btp:target-additional-information>  
3948 <btp:inferior-identifier>...hexstring...</btp:inferior-  
3949 identifier>  
3950 <btp:qualifiers> ?  
3951 ...qualifiers...  
3952 </btp:qualifiers>  
3953 </btp:confirm one phase>
```

REQUEST_PREPARE PREPARE INFERIORS

```
3955  
3956  
3957 <btp: request prepare-inferiors id?>  
3958 <btp:target-additional-information>  
3959 ...additional address information...  
3960 </btp:target-additional-information>  
3961 <btp:reply-address> ?  
3962 ...address...  
3963 </btp:reply-address>
```

```

3964 <u><btp:transaction-identifier>...hexstring...</btp:transaction-
3965 identifier> ?
3966 <u><btp:inferiors-list> ?
3967 <u><btp:inferior-handle>...hexstring...</btp:inferior-handle>
3968 +
3969 <u></btp:inferiors-list>
3970 <u><btp:qualifiers> ?
3971 <u>...qualifiers...
3972 <u></btp:qualifiers>
3973 <u></btp:request-prepare-inferiors>
3974
3975

```

REQUEST_CONFIRMCONFIRM_TRANSACTION

```

3976 <u><btp:confirm-transaction report-hazard="true|false" id?>
3977
3978 <u><btp:target-additional-information>
3979 <u>...additional address information...
3980 <u></btp:target-additional-information>
3981 <u><btp:reply-address>
3982 <u>...address...
3983 <u></btp:reply-address>
3984 <u><btp:transaction-identifier>...hexstring...</btp:transaction-
3985 identifier>
3986 <u><btp:inferiors-list> ?
3987 <u><btp:inferior-handle>...hexstring...</btp:inferior-handle>
3988 +
3989 <u></btp:inferiors-list>
3990 <u><btp:qualifiers> ?
3991 <u>...qualifiers...
3992 <u></btp:qualifiers>
3993 <u></btp:confirm-transaction>
3994
3995
3996

```

CONFIRM_COMPLETETRANSACTION_CONFIRMED

```

3997 <u><btp:transaction-confirmed-complete confirmed-
3998 received="true|false" id?>
3999 <u><btp:target-additional-information>
4000 <u>...additional address information...
4001 <u></btp:target-additional-information>
4002 <u><btp:decider-address> ?
4003 <u>...address...
4004 <u></btp:decider-address>
4005 <u><btp:transaction-identifier>...hexstring...</btp:transaction-
4006 identifier> ?
4007 <u><btp:qualifiers> ?
4008 <u>...qualifiers...
4009 <u></btp:qualifiers>
4010 <u></btp:transaction-confirmed-complete>
4011
4012
4013
4014

```

CANCEL_TRANSACTION

4015
4016
4017
4018
4019
4020
4021
4022
4023
4024
4025
4026
4027
4028
4029
4030

```
<btp:cancel_transaction id?>  
  <btp:target-additional-information>  
    ...additional address information...  
  </btp:target-additional-information>  
  <btp:reply-address> ?  
    ...address...  
  </btp:reply-address>  
  <btp:transaction-identifier>...hexstring...</btp:transaction-  
  identifier> ?  
  <btp:qualifiers> ?  
    ...qualifiers...  
  </btp:qualifiers>  
</btp:cancel_transaction>
```

REQUEST_CANCELCANCEL_INFERIORS

4031
4032
4033
4034
4035
4036
4037
4038
4039
4040
4041
4042
4043
4044
4045
4046
4047
4048
4049
4050

```
<btp:request-cancel-inferiors id?>  
  <btp:target-additional-information>  
    ...additional address information...  
  </btp:target-additional-information>  
  <btp:reply-address> ?  
    ...address...  
  </btp:reply-address>  
  <btp:transaction-identifier>...hexstring...</btp:transaction-  
  identifier> ?  
  <btp:inferiors-list> ?  
    <btp:inferior-handle>...hexstring...</btp:inferior-handle>  
  </btp:inferiors-list>  
  <btp:qualifiers> ?  
    ...qualifiers...  
  </btp:qualifiers>  
</btp:request-cancel-inferiors>
```

CANCEL_COMPLETETRANSACTION_CANCELLED

4051
4052
4053
4054
4055
4056
4057
4058
4059
4060
4061
4062
4063
4064
4065
4066

```
<btp:cancel-complete id?>  
  <btp:target-additional-information>  
    ...additional address information...  
  </btp:target-additional-information>  
  <btp:decider-address> ?  
    ...address...  
  </btp:decider-address>  
  <btp:transaction-identifier>...hexstring...</btp:transaction-  
  identifier> ?  
  <btp:qualifiers> ?  
    ...qualifiers...  
  </btp:qualifiers>  
</btp:cancel-complete>
```

4067
4068
4069
4070
4071
4072
4073
4074
4075
4076
4077
4078
4079
4080
4081
4082
4083
4084
4085
4086
4087
4088
4089
4090
4091
4092
4093
4094
4095
4096
4097
4098
4099
4100
4101
4102
4103
4104
4105
4106
4107
4108
4109
4110
4111
4112
4113
4114
4115
4116
4117
4118

REQUEST_STATUSES REQUEST_INFERIOR_STATUSES

```
<btp:request_statuses id?>
  <btp:target-additional-information>
    ...additional address information...
  </btp:target-additional-information>
  <btp:reply-address>
    ...address...
  </btp:reply-address>
  <btp:transactiontarget-
identifier>...hexstring...</btp:transactiontarget-identifier>
  <btp:inferiors-list> ?
    <btp:inferior-handle>...hexstring...</btp:inferior-handle>
  +
  </btp:inferiors-list>
  <btp:qualifiers> ?
    ...qualifiers...
  </btp:qualifiers>
</btp:request_statuses>
```

INFERIOR_STATUSES

```
<btp:inferior_statuses id?>
  <btp:target-additional-information>
    ...additional address information...
  </btp:target-additional-information>
  <btp:deciderresponders-address>
    ...address...
  </btp:respondersdecider-address>
  <btp:transactionresponders-
identifier>...hexstring...</btp:transactionresponders-identifier>
  <btp:status-list>
    <btp:status-item> +
      <btp:inferior-handle>...hexstring...</btp:inferior-
handle>
      <btp:status>active|resigned|preparing|prepared|
autonomously-confirmed|autonomously-cancelled|
confirming|confirmed|cancelling|cancelled|
cancel-contradiction|confirm-contradiction|
hazard</btp:status>
      <btp:qualifiers> ?
        ...qualifiers...
      </btp:qualifiers>
    </btp:status-item>
  </btp:status-list>
  <btp:qualifiers> ?
    ...qualifiers...
  </btp:qualifiers>
</btp:inferior_statuses>
```

4119
4120
4121
4122
4123
4124
4125
4126
4127
4128
4129
4130
4131
4132
4133
4134
4135
4136
4137
4138
4139
4140
4141
4142
4143
4144
4145
4146
4147
4148
4149
4150
4151
4152
4153
4154
4155
4156
4157
4158
4159
4160
4161
4162
4163
4164
4165
4166
4167
4168
4169

REQUEST_STATUS

```
<btp:request_status id?>
  <btp:target-additional-information>
    ...additional address information...
  </btp:target-additional-information>
  <btp:reply-address>
    ...address...
  </btp:reply-address>
  <btp:inferior<del>target</del>-
identifier>...hexstring...</btp:inferior<del>target</del>-identifier> ?
  <del><btp:transaction-identifier>...hexstring...</btp:transaction-
identifier></del> ?
  <btp:qualifiers> ?
    ...qualifiers...
  </btp:qualifiers>
</btp:request_status>
```

STATUS

```
<btp:status id?>
  <btp:target-additional-information>
    ...additional address information...
  </btp:target-additional-information>
  <btp:inferior<del>responder</del>-address><del> ?
    ...address...
  </btp:inferior<del>responder</del>-address>
  <btp:inferior<del>responder</del>-
identifier>...hexstring...</btp:inferior<del>responder</del>-identifier><del> ?
  <del><btp:decider-address> ?
  ...address...
  </btp:decider-address>
  <del><btp:transaction-identifier>...hexstring...</btp:transaction-
identifier></del> ?
  <btp:status-value> created|enrolling|active|resigning|
    resigned|preparing|prepared|
    confirming|confirmed|cancelling|cancelled|
    cancel-contradiction|confirm-contradiction|
    hazard|contradicted|unknown|inaccessible</btp:status-
value>
  <btp:qualifiers> ?
    ...qualifiers...
  </btp:qualifiers>
</btp:status>
```

REDIRECT

```
<btp:redirect id?>
```

```

4170 <del><btp:target-additional-information>
4171 <del>...additional-address-information...
4172 <del></btp:target-additional-information>
4173 <del><btp:superior-identifier>...hexstring...</btp:superior-
4174 <del>identifier> ?
4175 <del><btp:inferior-identifier>...hexstring...</btp:inferior-
4176 <del>identifier>
4177 <del><btp:old-address> +
4178 <del>...address...
4179 <del></btp:old-address>
4180 <del><btp:new-address> +
4181 <del>...address...
4182 <del></btp:new-address>
4183 <del><btp:qualifiers> ?
4184 <del>...qualifiers...
4185 <del></btp:qualifiers>
4186 <del></btp:redirect>
4187
4188

```

FAULT

```

4191 <btp:fault id?>
4192   <btp:target-additional-information>
4193     ...additional address information...
4194   </btp:target-additional-information>
4195   <btp:superior-identifier>...hexstring...</btp:superior-
4196 <del>identifier> ?
4197   <btp:inferior-identifier>...hexstring...</btp:inferior-
4198 <del>identifier> ?
4199   <btp:fault-type>...fault type name...</btp:fault-type>
4200   <btp:fault-data>...fault data...</btp:fault-data> ?
4201   <btp:qualifiers> ?
4202     ...qualifiers...
4203   </btp:qualifiers>
4204 </btp:fault>
4205
4206

```

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

```

4209
4210     o   general
4211     o   unknown-parameter
4212     o   wrong-state
4213     o   communication-failure
4214     o   invalid-superior
4215     o   duplicate-inferior
4216     o   unknown-inferior
4217

```

4218 Revisions of this specification may add other fault type names, which shall be simple strings
4219 of letters, numbers and hyphens. If other specifications define fault type names to be used
4220 with BTP, the names shall be URIs.

4221
4222 Fault data can take on various forms:

4223 Free text:

```
4224 <btp: fault-data>...string data...</btp: fault-data>
```

4225
4226 Identifier:

```
4227 <btp: fault-data>...hexstring...</btp: fault-data>
```

4228
4229 Inferior Identity:

```
4230 <btp: fault-data>  
4231 <btp: inferior-address> +  
4232 ...address...  
4233 </btp: inferior-address>  
4234 <btp: inferior-identifier>...hexstring...</btp: inferior-  
4235 identifier>  
4236 </btp: fault-data>
```

4237
4238

4239 **Standard qualifiers**

4240 The informal syntax for these messages assumes the namespace prefix “btpq” is associated
4241 with the URI “urn:oasis:names:tc:BTP:qualifiers”.

4242

4243 **Transaction timelimit**

```
4244 <btpq: transaction-timelimit>  
4245 <btpq: timelimit>  
4246 ...time in seconds...  
4247 </btpq: timelimit>  
4248 </btpq: transaction-timelimit>
```

4249

4250 **Inferior timeout**

```
4251 <btpq: inferior-timeout>  
4252 <btpq: timeout>  
4253 ...time in seconds...  
4254 </btpq: timeout>  
4255 <btpq: intended-decision>confirm|cancel</btpq: intended-decision>  
4256 </btpq: inferior-timeout>
```

4257

4258 **Minimum inferior timeout**

```
4259 <btpq: minimum-inferior-timeout>  
4260 <btpq: minimum-timeout>
```

4261

```
4267     ...time in seconds...
4268     </btpq:minimum-timeout>
4269 </btpq:minimum-inferior-timeout>
4270
```

4271 **Compounding of Messages**

4272
4273 Relating BTP to one another, in a “group” is represented by containing them within the
4274 btp:relatedgroup element, with the related messages as child elements. The processing for the
4275 group is defined in the section ‘Groups – combinations of related messages’. For example

```
4276  
4277     <btp:relatedgroup>
4278     <btp:context-reply>
4279     ...<completion-status>related</completion-status> ...
4280     </btp:context-reply>
4281     <btp:enrol>...</btp:enrol>
4282     <btp:prepared>...</btp:prepared>
4283     </btp:relatedgroup>
4284
```

4285 If the rules for the group state that the target address of the abstract message is omitted, the
4286 corresponding target-address-information element shall be absent in the message in the
4287 relatedgroup. The carrier protocol binding specifies how a relation between application and
4288 BTP messages is represented.

4289
4290 Bundling (semantically insignificant combination) of BTP messages and related groups is
4291 indicated with the "btp:messages" element, with the bundled messages and related groups as
4292 child elements. For example (confirming one and cancelling another inferiors of a cohesion):

```
4293  
4294     <btp:messages>
4295     <btp:enrolconfirm>...</btp:enrolconfirm>
4296     <btp:preparedcancel>...</btp:preparedcancel>
4297     </btp:messages>
4298
```

4299
4300 Relating BTP messages to one another is achieved through containment. For example:

```
4301  
4302     <btp:context-reply>
4303     ...<completion-status>related</completion-status>...
4304     <btp:enrol>...</btp:enrol>
4305     </btp:context-reply>
4306
```

4307
4308 The carrier protocol binding specifies how a relation between application and BTP messages
4309 is represented.

4310

4311 Carrier Protocol Bindings

4312

4313 The notion of bindings is introduced to act as the glue between the BTP ~~XML~~ messages and
4314 an underlying transport. A binding specification must define various particulars of how the
4315 BTP messages are carried and some aspects of how the related application messages are
4316 carried. This document specifies two bindings: a SOAP binding and a SOAP + Attachments
4317 binding. However, other bindings could be specified by the Oasis BTP technical committee
4318 or by a third party. For example, in the future a binding might exist to put a BTP message
4319 directly on top of HTTP without the use of SOAP, or a closed community could define their
4320 own binding. To ensure that such specifications are complete, the Binding Proforma defines
4321 the information that must be included in a binding specification.
4322

4323 Carrier Protocol Binding Proforma

4324

4325 A BTP carrier binding specification should provide the following information:

4326

4327 **Binding name:** A name for the binding, as used in the “binding name” field of BTP
4328 addresses (and available for declaring the capabilities of an implementation). Binding
4329 specified in this document, and future revisions of this document have binding names that are
4330 simple strings of letters, numbers and hyphens (and, in particular, do not contain colons).
4331 Bindings specified elsewhere shall have binding names that are URIs. Bindings specified in
4332 this document use numbers to identify the version of the binding, not the version(s) of the
4333 carrier protocol.

4334

4335 **Binding address format:** This section states the format of the “binding address” field of a
4336 BTP address for this binding. For many bindings, this will be a URL of some kind; for other
4337 bindings it may be some other form

4338

4339 **BTP message representation:** This section will define how BTP messages are represented.
4340 For many bindings, the BTP message syntax will be ~~this will be the normal string encoding~~
4341 of the XML, in accordance as specified in ~~with~~ the XML schema defined in this document,
4342 and the normal string encoding of that XML will be used.

4343

4344 **Mapping for BTP messages (unrelated) :** This section will define how BTP messages that
4345 are not related to application messages are sent in either direction between Superior and
4346 Inferior. (i.e. those messages sent directly between BTP actors). This mapping need not be
4347 symmetric (i.e. Superior to Inferior may differ to some degree to Inferior to Superior). The
4348 mapping may define particular rules for particular BTP messages, or messages with particular
4349 parameter values (e.g. the FAULT message with “fault-type” “CommunicationFailure” will
4350 typically not be sent as a BTP message). The mapping states any constraints or requirements
4351 on which BTP may or must be bundled together by compounding.

4352

4353 **Mapping for BTP messages related to application messages-** This section will define
4354 how BTP messages that are related to application messages are sent. A binding specification
4355 may defer details of this to a particular application (e.g. a mapping specification could just

4356 say “the CONTEXT may be carried as a parameter of an application invocation”).
4357 Alternatively, the binding may specify a general method that represents the relationship
4358 between application and BTP messages.
4359

4360 **Implicit messages:** This section specifies which BTP messages, if any, are not sent explicitly
4361 but are treated as implicit in application messages or other BTP messages. This may depend
4362 on particular parameter values of the BTP messages or the application messages.
4363

4364 **Faults:** The relationship between the fault and exception reporting mechanisms of the carrier
4365 protocol and of BTP shall be defined. This may include definition of which carrier protocol
4366 exceptions are equivalent to a FAULT/communication-failure message.
4367

4368 **Relationship to other bindings:** Any relationship to other bindings is defined in this section.
4369 If BTP addresses with different bindings are be considered to match (for purposes of
4370 identifying the peer Superior/Inferior and redirection), this should be specified here.
4371

4372 **Limitations on BTP use:** Any limitations on the full range of BTP functionality that are
4373 imposed by use of this binding should be listed. This would include limitations on which
4374 messages can be sent, which event sequences are supported and restrictions on parameter
4375 values. Such limitations may reduce the usefulness of an implementation, but may be
4376 appropriate in certain environments.
4377

4378 **Other:** Other features of the binding, especially any that will potentially affect interoperability
4379 should be specified here. This may include restrictions or requirements on the use or support
4380 of optional carrier parameters or mechanisms➤.
4381

4382 Bindings for request/response carrier protocols

4383
4384 BTP does not generally follow request/response pattern. In particular, on the outcome
4385 relationship either side may initiate a message – this is an essential part of the presume-abort
4386 recovery paradigm although it is not limited to recovery cases. However, there are some BTP
4387 messages, especially in the control relationship, that do have a request/response pattern.
4388 Many (potential) carrier protocols (e.g. HTTP) do have a request/response pattern. The
4389 specification of a binding specification to a request/response carrier protocol needs to state
4390 what rules apply – which messages can be carried by requests, which by responses. The
4391 simplest rule is to send all BTP messages on requests, and let the carrier responses travel back
4392 empty. This would be inefficient in use of network resources, and possibly inconvenient
4393 when used for the BTP request/response pairs.
4394

4395 This section defines a set of rules that allow more efficient use of the carrier, while allowing
4396 the initiator of a BTP request/response pair to ensure the BTP response is sent back on the
4397 carrier response. These rules are specified in this section to enable binding specifications to
4398 reference them, without requiring each binding specification to repeat similar information.
4399

4400 A binding to a request/response carrier is not required to use these rules. It may define other
4401 rules.
4402

Request/response exploitation rules

These rules allow implementations to use the request and response of the carrier protocol efficiently, and, when a BTP request/response exchange occurs, to either treat the request/response exchanges of the carrier protocol and of BTP independently, if both sides wish, or allow either side to map them closely.

Under these rules, an implementation sending a BTP request (i.e. a message, other than CONTEXT, which has “reply-address” as a parameter in the abstract message definition), can ensure that it and the reply map to a carrier request/response by supplying no value for the “reply-address”. An implementation receiving such a request is required to send the BTP response on the carrier response.

Conversely, if an implementation does supply a “reply-address” value on the request, the receiver has the option of sending the BTP response back on the carrier response, or sending it on a new carrier request.

Within the outcome relationship, apart from ENROL/ENROLLED, there is no “reply-address”, and the parties know each other’s “address-as-superior” and “address-as-inferior”. Both sides are permitted to treat the carrier request/response exchanges as just opportunities for sending messages to the appropriate destination.

The rules:

- a) A BTP actor **may** bundle one or more BTP messages and related groups that have the same binding address for their target in a single btp:messages and transmit this btp:messages element on a carrier protocol request. There is no restriction on which combinations of messages and groups may be so bundled other than that they have the same binding address, and that this binding address is usable as the destination of a carrier protocol request.
- b) A BTP actor that has received a carrier protocol request to which it has not yet responded, and which has one or more BTP messages and groups whose binding address for the target matches the origin of the carrier request **may** bundle such BTP messages in a single btp:messages element and transmit that on the carrier protocol response.
- c) A BTP actor that has received, on a carrier protocol request, one or more BTP messages or related groups that require a BTP response and for which no reply address was supplied, **must** bundle the responding BTP message and groups in a btp:messages element and transmit this element on the carrier protocol response to the request that carried the BTP request.
- d) Where only one message or group is to be sent, it shall be contained within a btp:messages element, as a bundle of one element.

- 4449 e) A BTP actor that receives a carrier protocol request carrying BTP messages that
4450 do have a reply address, or which initiate processing that produces BTP messages
4451 whose target binding address matches the origin of the request, **may** freely
4452 choose whether to use the carrier protocol response for the replies, or to send
4453 back an “empty carrier protocol response”, and send the BTP replies in a
4454 separately initiated carrier protocol request. The characteristics of an “empty
4455 carrier protocol response” shall be stated in the particular binding specification.
4456
4457 f) A BTP actor that sends BTP messages on a carrier protocol request **must** be able
4458 to accept returning BTP messages on the corresponding carrier protocol response
4459 and, if the actor has offered an address on which it will receive carrier requests,
4460 must be able to accept “replying” BTP messages on a separate carrier protocol
4461 request.
4462

4463 SOAP Binding

4464
4465 This binding describes how BTP messages will be carried using SOAP as in the [SOAP 1.1](#)
4466 [specification, using the SOAP literal messaging style conventions.](#) If no application message
4467 is sent at the same time, the BTP messages are contained within the SOAP Body element. If
4468 application messages are sent, the BTP messages are contained in the SOAP Header element.
4469

4470 **Binding name:** soap-http-1

4471
4472 **Binding address format:** shall be a URL, of type HTTP.

4473
4474 **BTP message representation:** The string representation of the XML, as specified in the
4475 XML schema defined in this document shall be used. The BTP XML messages are embedded
4476 in the SOAP message without the use of any specific encoding rules (literal style SOAP
4477 message); hence the encodingStyle attribute need not be set or can be set to an empty
4478 string, conform to the rules of the Section 5 (of the SOAP 1.1 specification) SOAP Encoding
4479 as specified by the URI: "http://schemas.xmlsoap.org/soap/encoding/".
4480

4481 **Mapping for BTP messages (unrelated):** The “request/response exploitation” rules shall be
4482 used.

4483
4484 BTP messages sent on an HTTP request or HTTP response which is not carrying an
4485 application message, the messages are contained in a single btp:messages element which is
4486 the immediate child element of the SOAP Body element.

4487
4488 An “empty carrier protocol response” sent after receiving an HTTP request containing a
4489 btp:messages element in the SOAP Body and the implementation BTP actor chooses just to
4490 reply at the lower level (and when the request/response exploitation rules allow an empty
4491 carrier protocol response), shall be any of:

- 4492 a) an empty HTTP response
4493 b) an HTTP response containing an empty SOAP Envelope
4494 c) an HTTP response containing a SOAP Envelope containing a single, empty
4495 btp:messages element.

4496
4497
4498
4499
4500
4501
4502
4503
4504
4505
4506
4507
4508
4509
4510
4511
4512
4513
4514
4515
4516
4517
4518

4519
4520
4521

4522
4523
4524
4525
4526
4527

4528
4529
4530

4531
4532

4533
4534
4535
4536
4537
4538

The receiver (the initial sender of the HTTP request) shall treat these in the same way – they have no effect on the BTP sequence (other than indicating that the earlier sending did not cause a communication failure.)

~~If no application message is being sent at the same time, BTP messages shall be contained in a `btm:messages` element which shall be an immediate child element of the SOAP-Body. There shall be precisely one `btm:messages` element. Any number of BTP messages with the same binding address in their target address may be carried in the same `btm: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` and `ENROL` 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 `btm:messages` element in the `SOAP_Header`. There shall be precisely one `btm:messages` element in the `SOAP_Header`.

The “request/response exploitation” rules shall apply to the BTP messages carried in the SOAP Header, as if they had been carried in a SOAP Body, unrelated to an application message, sent to the same binding address.

Note – The application protocol itself (which is using the SOAP Body) may use the SOAP RPC or document approach – this is determined by the application.

Only `CONTEXT` and `ENROL` messages are related (&) to application messages. If there is only one `CONTEXT` or one `ENROL` message present in the SOAP Header, it is assumed to be related to the whole of the application message in the SOAP Body. If there are multiple `CONTEXT` or `ENROL` messages, any relation of these BTP messages shall be indicated by application specific means.

Note 1 – An application protocol could use references to the ID values of the BTP messages to indicate relation between BTP `CONTEXT` or `ENROL` messages and the application message.

Note 2 -- However indicated, what the relatedness means, or even whether it has any significance at all, is a matter for the application.

Implicit messages: A SOAP ~~fault~~`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.

4539 **Faults:** A SOAP ~~fault~~ **FAULT** or other communication failure shall be treated as
4540 FAULT/communication-failure.

4541
4542 **Relationship to other bindings:** A BTP address for Superior or Inferior that has the binding
4543 string “soap-http-1” is considered to match one that has the binding string “soap-attachments-
4544 http-1” if the binding address and additional information fields match.

4545
4546 **Limitations on BTP use:** None

4547
4548 **Other:** The SOAP BTP binding does not make use of SOAPAction HTTP header or actor
4549 attribute. The SOAPAction HTTP header is left to be application specific when there are
4550 application messages in the SOAP ~~Body~~, as an already existing web service that is being
4551 upgraded to use BTP might have already made use of SOAPAction. The SOAPAction HTTP
4552 header shall be omitted when the SOAP message carries only BTP messages in the SOAP
4553 ~~Body~~.

4554
4555 The SOAP mustUnderstand attribute, when used on the btp:messages containing a ~~the~~ BTP
4556 CONTEXT, ensures that the ~~receiver~~ (server, ~~as a whole~~) ~~supports BTP sufficiently to~~
4557 ~~determine~~ whether any enrolments are necessary and ~~replies~~ with CONTEXT_REPLY as
4558 appropriate. ~~The sender of the CONTEXT (and related application message) can use this to~~
4559 ~~ensure that the application work is performed as part of the business transaction, assuming the~~
4560 ~~receiver’s SOAP implementation supports the mustUnderstand attribute.~~ If mustUnderstand if
4561 false, a ~~server-receiver~~ can ignore the CONTEXT (if BTP is not supported there), ~~and no~~
4562 ~~CONTEXT_REPLY will be returned.~~ It is a ~~local implementation or configuration~~ option ~~on~~
4563 ~~the sender (client) side~~ whether ~~the absence of a CONTEXT_REPLY is assumed to be~~
4564 ~~equivalent to a CONTEXT_REPLY/ok (and the business transaction allowed to proceed to~~
4565 ~~confirmation).~~ ~~is assumed to be implicit in the HTTP response in such a case. (If no~~
4566 ~~CONTEXT_REPLY/ok is assumed, it will be impossible for the business transaction to~~
4567 ~~confirm).~~

4569 Note – some SOAP implementations may not support the mustUnderstand
4570 attribute sufficiently to enforce these requirements. ~~If using such an~~
4571 ~~implementation on the service side, it may be necessary to assume a~~
4572 ~~CONTEXT_REPLY/ok.~~

4573 4574 **Example scenario using SOAP binding**

4575
4576 The example below shows an application request with CONTEXT message sent from
4577 client.example.com (which includes the Superior) to services.example.com (Service).

```
4578  
4579  
4580 <soap:Envelope  
4581     xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"  
4582     soap-env encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
```

```

4585 <soap:Header>
4586
4587     <btp:messages xmlns:btp="urn:oasis:names:tc:BTP:xml">
4588         <btp:context superior-type="atom">
4589             <btp:superior-address>
4590                 <btp:binding>soap-http-1</btp:binding>
4591                 <btp:binding-
4592 address>http://client.example.com/soaphandler</btp:binding-
4593 address>
4594                 <btp:additional-information>btpengine</btp:additional-
4595 information>
4596             </btp:superior-address>
4597             <btp:superior-identifier>1001</btp:superior-identifier>
4598             <btp:qualifiers>
4599                 <btpq:transaction-timelimit
4600 xmlns:btpq="urn:oasis:names:tc:BTP:qualifiers">1800</btpq:transact
4601 ion-timelimit>
4602             </btp:qualifiers>
4603         </btp:context>
4604     </btp:messages>
4605
4606 </soap:Header>
4607
4608 <soap:Body>
4609
4610     <ns1:orderGoods
4611 xmlns:ns1="http://example.com/2001/Services/xyzgoods">
4612         <custID>ABC8329045</custID>
4613         <itemID>224352</itemID>
4614         <quantity>5</quantity>
4615     </ns1:orderGoods>
4616
4617 </soap:Body>
4618
4619 </soap:Envelope>
4620

```

The example below shows CONTEXT_REPLY and a related ~~(and therefore contained)~~ ENROL message sent from services.example.com to client.example.com, in reply to the previous message. There is no application response, so the BTP messages are in the SOAP Body. The ENROL message does not contain the target-additional-information, since the grouping rules for CONTEXT_REPLY & ENROL omit the target address (the receiver of this example remembers the superior address from the original CONTEXT)

```

4621
4622
4623
4624
4625
4626
4627
4628
4629 <soap:Envelope
4630     xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
4631     soap-
4632 env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
4633
4634     <soap:Header>
4635     </soap:Header>
4636

```

4637
4638
4639
4640
4641
4642
4643
4644
4645
4646
4647
4648
4649
4650
4651
4652
4653
4654
4655
4656
4657
4658
4659
4660
4661
4662
4663
4664
4665
4666
4667
4668
4669
4670
4671
4672
4673
4674
4675
4676
4677
4678
4679
4680
4681
4682

```
<soap:Body>
  <btp:messages xmlns:btp="urn:oasis:names:tc:BTP:xml">
    <btp:relatedgroup>
      <btp:context-reply>
        <btp:superior-address>
          <btp:binding>soap-http-1</btp:binding>
          <btp:binding-address>
            http://client.example.com/soaphandler
          </btp:binding-address>
          <btp:additional-information>
            btpengine
          </btp:additional-information>
        </btp:superior-address>
        <btp:superior-identifier>1001</btp:superior-identifier>
        <completion-status>related</completion-status>
      </btp:context-reply>
      <btp:enrol reply-requested="false">
        <del>btp:target-additional-information</del>
        btpengine
      </del>
      <btp:superior-identifier>
        1001
      </btp:superior-identifier>
      <btp:inferior-address>
        <btp:binding>soap-http-1</btp:binding>
        <btp:binding-address>
          http://services.example.com/soaphandler
        </btp:binding-address>
        </btp:inferior-address>
        <btp:inferior-identifier>
          AAAB
        </btp:inferior-identifier>
      </btp:enrol>
    </del>
  </btp:relatedgroup>
</btp:messages>
</soap:Body>
</soap:Envelope>
```

SOAP + Attachments Binding

4683
4684
4685
4686
4687
4688

This binding describes how BTP messages will be carried using SOAP as in the [SOAP Messages with Attachments](#) specification. It is a superset of the Basic SOAP binding, soap-http-1. The two bindings only differ when application messages are sent.

4689 **Binding name:** soap-attachments-http-1
4690
4691 **Binding address format:** as for soap-http-1
4692
4693 **BTP message representation:** As for soap-http-1
4694
4695 **Mapping for BTP messages (unrelated):** As for “soap-http-1”, except the SOAP:Envelope
4696 containing the SOAP:Body containing the BTP messages shall be in a MIME body part, as
4697 specified in [SOAP Messages with Attachments](#) specification. If an application message is
4698 being sent at the same time, the mapping for related messages for this binding shall be used,
4699 as if the BTP messages were related to the application message(s).
4700
4701 **Mapping for BTP messages related to application messages:** MIME packaging shall be
4702 used. One of the MIME multipart/related parts shall contain a SOAP:Envelope, whose SOAP
4703 :Headers element shall contain precisely one bpm:messages element, containing any BTP
4704 messages. Any BTP CONTEXT in the bpm:messages is considered to be related to the
4705 application message(s) in the SOAP:Body, and to also any of the MIME parts referenced
4706 from the SOAP:Body (using the “href” attribute).
4707
4708 **Implicit messages:** As for soap-http-1.
4709
4710 **Faults:** As for soap-http-1.
4711
4712 **Relationship to other bindings:** A BTP address for Superior or Inferior that has the binding
4713 string “soap-http-1” is considered to match one that has the binding string “soap-
4714 attachments-http-1” if the binding address and additional information fields match.
4715
4716 **Limitations on BTP use:** None
4717
4718 **Other:** As for soap-http-1

4719
4720 *Example using SOAP + Attachments binding*

```
4721  
4722 MIME-Version: 1.0  
4723 Content-Type: Multipart/Related; boundary=MIME_boundary;  
4724 type=text/xml;  
4725         start="someID"  
4726  
4727 --MIME_boundary  
4728 Content-Type: text/xml; charset=UTF-8  
4729 Content-ID: someID  
4730  
4731 <?xml version='1.0' ?>  
4732 <soap:Envelope  
4733     xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"  
4734     soap-  
4735     env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
```

```

4737 <soap:Header>
4738
4739     <btp:messages xmlns:btp="urn:oasis:names:tc:BTP:xml">
4740         <btp:context superior-type="atom">
4741             <btp:superior-address>
4742                 <btp:binding>soap-http-1</btp:binding>
4743                 <btp:binding-address>
4744                     http://client.example.com/soaphandler
4745                 </btp:binding-address>
4746             </btp:superior-address>
4747             <btp:superior-identifier>1001</btp:superior-identifier>
4748         </btp:context>
4749     </btp:messages>
4750
4751 </soap:Header>
4752
4753 <soap:Body>
4754     <orderGoods href="cid:anotherID"/>
4755 </soap:Body>
4756
4757 </soap:Envelope>
4758
4759 --MIME_boundary
4760 Content-Type: text/xml
4761 Content-ID: anotherID
4762
4763     <ns1:orderGoods
4764 xmlns:ns1="http://example.com/2001/Services/xyzgoods">
4765         <custID>ABC8329045</custID>
4766         <itemID>224352</itemID>
4767         <quantity>5</quantity>
4768     </ns1:orderGoods>
4769
4770
4771 --MIME_boundary--
4772
4773

```

XML Schema ~~for SOAP Bindings~~

```

4774
4775
4776 <?xml version="1.0"?>
4777 <schema targetNamespace="urn:oasis:names:tc:BTP:xml"
4778     xmlns="http://www.w3.org/2001/XMLSchema"
4779     xmlns:tns="urn:oasis:names:tc:BTP:xml">
4780
4781     <complexType name="qualifier_type">
4782         <simpleContent>
4783             <extension base="string">
4784                 <attribute name="must-be-understood" type="boolean"/>
4785                 <attribute name="to-be-propagated" type="boolean"/>
4786             </extension>
4787         </simpleContent>
4788     </complexType>

```

```

4789 <element name="qualifier" type="tns:qualifier_type"/>
4790 <element name="qualifiers">
4791   <complexType>
4792     <sequence>
4793       <element ref="tns:qualifier" maxOccurs="unbounded"/>
4794     </sequence>
4795   </complexType>
4796 </element>
4797
4798 <complexType name="address">
4799   <sequence>
4800     <element name="binding-name" type="string"/>
4801     <element name="binding-address" type="string"/>
4802     <element name="additional-information" type="string"
4803 minOccurs="0"/>
4804   </sequence>
4805 </complexType>
4806
4807 <simpleType name="identifier">
4808   <restriction base="string">
4809     <pattern value="([0-9,A-Z])*"/>
4810   </restriction>
4811 </simpleType>
4812
4813 <simpleType name="superior-type">
4814   <restriction base="string">
4815     <enumeration value="cohesion"/>
4816     <enumeration value="atom"/>
4817   </restriction>
4818 </simpleType>
4819
4820 <simpleType name="transaction-type">
4821   <restriction base="string">
4822     <enumeration value="cohesion"/>
4823     <enumeration value="atom"/>
4824   </restriction>
4825 </simpleType>
4826
4827
4828 <element name="context">
4829   <complexType>
4830     <sequence>
4831       <element name="superior-address" type="tns:address"
4832 maxOccurs="unbounded"/>
4833       <element name="superior-identifier" type="tns:identifier"/>
4834       <element ref="tns:qualifiers" minOccurs="0"/>
4835     </sequence>
4836     <attribute name="id" type="ID" use="optional"/>
4837     <attribute name="superior-type" type="tns:superior-type"
4838 use="required"/>
4839   </complexType>
4840 </element>
4841

```

```

4842     <element name="context-reply">
4843         <complexType>
4844             <sequence>
4845                 <element name="superior-address" type="tns:address"
4846 maxOccurs="unbounded"/>
4847                 <element name="superior-identifier" type="tns:identifier"/>
4848                 <element name="completion-status">
4849                     <simpleType>
4850                         <restriction base="string">
4851                             <enumeration value="completed"/>
4852                             <enumeration value="related"/>
4853                             <enumeration value="repudiated"/>
4854                         </restriction>
4855                     </simpleType>
4856                 </element>
4857                 <element ref="tns:qualifiers" minOccurs="0"/>
4858             </sequence>
4859             <attribute name="id" type="ID"/>
4860             <attribute name="superior-type" type="tns:superior-type"
4861 use="required"/>
4862         </complexType>
4863     </element>
4864
4865     <element name="begin">
4866         <complexType>
4867             <sequence>
4868                 <element name="target-additional-information"
4869 type="string"/>
4870                 <element name="reply-address" type="tns:address"/>
4871                 <element ref="tns:qualifiers" minOccurs="0"/>
4872             </sequence>
4873             <attribute name="id" type="ID"/>
4874             <attribute name="transaction-type" type="tns:superior-type"
4875 use="required"/>
4876         </complexType>
4877     </element>
4878
4879     <element name="begun">
4880         <complexType>
4881             <sequence>
4882                 <element name="target-additional-information"
4883 type="string"/>
4884                 <element name="decider-address" type="tns:address"
4885 minOccurs="0"/>
4886                 <element name="transaction-identifier"
4887 type="tns:identifier" minOccurs="0"/>
4888                 <element name="inferior-handle" type="tns:identifier"
4889 minOccurs="0"/>
4890                 <element name="inferior-address" type="tns:address"
4891 minOccurs="0"/>
4892                 <element ref="tns:qualifiers" minOccurs="0"/>
4893             </sequence>
4894             <attribute name="id" type="ID"/>

```

```

4895         <attribute name="transaction-type" type="tns:superior-type"
4896 use="required"/>
4897     </complexType>
4898 </element>
4899
4900     <element name="enrol">
4901         <complexType>
4902             <sequence>
4903                 <element name="target-additional-information"
4904 type="string"/>
4905                 <element name="superior-identifier" type="tns:identifier"/>
4906                 <element name="reply-address" type="tns:address"
4907 minOccurs="0"/>
4908                 <element name="inferior-address" type="tns:address"
4909 minOccurs="1" maxOccurs="unbounded"/>
4910                 <element name="inferior-identifier" type="tns:identifier"/>
4911                 <element ref="tns:qualifiers" minOccurs="0"/>
4912             </sequence>
4913             <attribute name="id" type="ID"/>
4914             <attribute name="reply-requested" type="boolean"/>
4915         </complexType>
4916 </element>
4917
4918
4919     <element name="enrolled">
4920         <complexType>
4921             <sequence>
4922                 <element name="target-additional-information"
4923 type="string"/>
4924                 <element name="inferior-identifier" type="tns:identifier"/>
4925                 <element name="inferior-handle" type="tns:identifier"
4926 minOccurs="0"/>
4927                 <element ref="tns:qualifiers" minOccurs="0"/>
4928             </sequence>
4929             <attribute name="id" type="ID"/>
4930         </complexType>
4931 </element>
4932
4933     <element name="resign">
4934         <complexType>
4935             <sequence>
4936                 <element name="target-additional-information"
4937 type="string"/>
4938                 <element name="superior-identifier" type="tns:identifier"/>
4939                 <element name="inferior-address" type="tns:address"
4940 minOccurs="1" maxOccurs="unbounded"/>
4941                 <element name="inferior-identifier" type="tns:identifier"/>
4942                 <element ref="tns:qualifiers" minOccurs="0"/>
4943             </sequence>
4944             <attribute name="id" type="ID"/>
4945             <attribute name="response-requested" type="boolean"/>
4946         </complexType>
4947 </element>

```

```

4948
4949     <element name="resigned">
4950         <complexType>
4951             <sequence>
4952                 <element name="target-additional-information"
4953 type="string"/>
4954                 <element name="inferior-identifier" type="tns:identifier"/>
4955                 <element ref="tns:qualifiers" minOccurs="0"/>
4956             </sequence>
4957             <attribute name="id" type="ID"/>
4958         </complexType>
4959     </element>
4960
4961     <element name="prepare">
4962         <complexType>
4963             <sequence>
4964                 <element name="target-additional-information"
4965 type="string"/>
4966                 <element name="inferior-identifier" type="tns:identifier"
4967 minOccurs="0"/>
4968                 <element name="reply-address" type="tns:address"
4969 minOccurs="0"/>
4970                 <element name="transaction-identifier"
4971 type="tns:identifier" minOccurs="0"/>
4972                 <element name="inferiors-list" minOccurs="0">
4973                     <complexType>
4974                         <sequence>
4975                             <element name="inferior-handle"
4976 type="tns:identifier" maxOccurs="unbounded"/>
4977                         </sequence>
4978                     </complexType>
4979                 </element>
4980                 <element ref="tns:qualifiers" minOccurs="0"/>
4981             </sequence>
4982             <attribute name="id" type="ID"/>
4983         </complexType>
4984     </element>
4985
4986     <element name="prepared">
4987         <complexType>
4988             <sequence>
4989                 <element name="target-additional-information"
4990 type="string"/>
4991                 <element name="superior-identifier" type="tns:identifier"/>
4992                 <element name="inferior-address" type="tns:address"
4993 maxOccurs="unbounded"/>
4994                 <element name="inferior-identifier" type="tns:identifier"/>
4995                 <element ref="tns:qualifiers" minOccurs="0"/>
4996             </sequence>
4997             <attribute name="id" type="ID"/>
4998             <attribute name="default-is-cancel" type="boolean"/>
4999         </complexType>
5000     </element>

```

```

5001
5002     <element name="confirm">
5003         <complexType>
5004             <sequence>
5005                 <element name="target-additional-information"
5006 type="string"/>
5007                 <element name="inferior-identifier" type="tns:identifier"/>
5008                 <element ref="tns:qualifiers" minOccurs="0"/>
5009             </sequence>
5010             <attribute name="id" type="ID"/>
5011         </complexType>
5012     </element>
5013
5014     <element name="confirmed">
5015         <complexType>
5016             <sequence>
5017                 <element name="target-additional-information"
5018 type="string"/>
5019                 <element name="superior-identifier" type="tns:identifier"/>
5020                 <element name="inferior-address" type="tns:address"
5021 minOccurs="0"/>
5022                 <element name="inferior-identifier" type="tns:identifier"
5023 minOccurs="0"/>
5024                 <element name="decider-address" type="tns:address"
5025 minOccurs="0"/>
5026                 <element name="transaction-identifier"
5027 type="tns:identifier" minOccurs="0"/>
5028                 <element ref="tns:qualifiers" minOccurs="0"/>
5029             </sequence>
5030             <attribute name="id" type="ID"/>
5031             <attribute name="confirmed-received" type="boolean"/>
5032         </complexType>
5033     </element>
5034
5035     <element name="cancel">
5036         <complexType>
5037             <sequence>
5038                 <element name="target-additional-information"
5039 type="string"/>
5040                 <element name="inferior-identifier" type="tns:identifier"
5041 minOccurs="0"/>
5042                 <element name="reply-address" type="tns:address"
5043 minOccurs="0"/>
5044                 <element name="transaction-identifier"
5045 type="tns:identifier" minOccurs="0"/>
5046                 <element name="decider-address" type="tns:address"
5047 minOccurs="0"/>
5048                 <element name="transaction-identifier"
5049 type="tns:identifier" minOccurs="0"/>
5050                 <element name="inferiors-list" minOccurs="0">
5051                     <complexType>
5052                         <sequence>

```

```

5053         <element name="inferior-handle"
5054 type="tns:identifier" maxOccurs="unbounded"/>
5055         </sequence>
5056     </complexType>
5057 </element>
5058     <element ref="tns:qualifiers" minOccurs="0"/>
5059 </sequence>
5060     <attribute name="id" type="ID"/>
5061 </complexType>
5062 </element>
5063
5064     <element name="cancelled">
5065         <complexType>
5066             <sequence>
5067                 <element name="target-additional-information"
5068 type="string"/>
5069                 <element name="superior-identifier" type="tns:identifier"/>
5070                 <element name="inferior-address" type="tns:address"
5071 maxOccurs="unbounded"/>
5072                 <element name="inferior-identifier" type="tns:identifier"
5073 minOccurs="0"/>
5074                 <element name="decider-address" type="tns:address"
5075 minOccurs="0"/>
5076                 <element name="transaction-identifier"
5077 type="tns:identifier" minOccurs="0"/>
5078                 <element ref="tns:qualifiers" minOccurs="0"/>
5079             </sequence>
5080             <attribute name="id" type="ID"/>
5081         </complexType>
5082     </element>
5083
5084     <element name="hazard">
5085         <complexType>
5086             <sequence>
5087                 <element name="target-additional-information"
5088 type="string"/>
5089                 <element name="superior-identifier" type="tns:identifier"/>
5090                 <element name="inferior-address" type="tns:address"
5091 maxOccurs="unbounded"/>
5092                 <element name="inferior-identifier" type="tns:identifier"/>
5093                 <element ref="tns:qualifiers" minOccurs="0"/>
5094             </sequence>
5095             <attribute name="id" type="ID"/>
5096         </complexType>
5097     </element>
5098
5099     <element name="contradiction">
5100         <complexType>
5101             <sequence>
5102                 <element name="target-additional-information"
5103 type="string"/>
5104                 <element name="inferior-identifier" type="tns:identifier"/>
5105                 <element ref="tns:qualifiers" minOccurs="0"/>

```

```

5106         </sequence>
5107         <attribute name="id" type="ID"/>
5108     </complexType>
5109 </element>
5110
5111     <element name="superior-state">
5112         <complexType>
5113             <sequence>
5114                 <element name="target-additional-information"
5115 type="string"/>
5116                 <element name="inferior-identifier" type="tns:identifier"/>
5117                 <element name="status">
5118                     <simpleType>
5119                         <restriction base="string">
5120                             <enumeration value="active"/>
5121                             <enumeration value="prepared-received"/>
5122                             <enumeration value="inaccessible"/>
5123                             <enumeration value="unknown"/>
5124                         </restriction>
5125                     </simpleType>
5126                 </element>
5127                 <element ref="tns:qualifiers" minOccurs="0"/>
5128             </sequence>
5129             <attribute name="id" type="ID"/>
5130             <attribute name="reply-requested" type="boolean"/>
5131         </complexType>
5132 </element>
5133
5134     <element name="inferior-state">
5135         <complexType>
5136             <sequence>
5137                 <element name="target-additional-information"
5138 type="string"/>
5139                 <element name="superior-identifier" type="tns:identifier"/>
5140                 <element name="inferior-address" type="tns:address"
5141 maxOccurs="unbounded"/>
5142                 <element name="inferior-identifier" type="tns:identifier"/>
5143                 <element name="status">
5144                     <simpleType>
5145                         <restriction base="string">
5146                             <enumeration value="active"/>
5147                             <enumeration value="prepared-received"/>
5148                             <enumeration value="inaccessible"/>
5149                             <enumeration value="unknown"/>
5150                         </restriction>
5151                     </simpleType>
5152                 </element>
5153                 <element ref="tns:qualifiers" minOccurs="0"/>
5154             </sequence>
5155             <attribute name="id" type="ID"/>
5156             <attribute name="reply-requested" type="boolean"/>
5157         </complexType>
5158 </element>

```

```

5159
5160     <element name="confirm-one-phase">
5161         <complexType>
5162             <sequence>
5163                 <element name="target-additional-information"
5164 type="string"/>
5165                 <element name="inferior-identifier" type="tns:identifier"/>
5166                 <element ref="tns:qualifiers" minOccurs="0"/>
5167             </sequence>
5168             <attribute name="id" type="ID"/>
5169             <attribute name="report-hazard" type="boolean"/>
5170         </complexType>
5171     </element>
5172
5173     <element name="request-confirm">
5174         <complexType>
5175             <sequence>
5176                 <element name="target-additional-information"
5177 type="string"/>
5178                 <element name="reply-address" type="tns:address"/>
5179                 <element name="transaction-identifier"
5180 type="tns:identifier"/>
5181                 <element name="inferiors-list" minOccurs="0">
5182                     <complexType>
5183                         <sequence>
5184                             <element name="inferior-handle"
5185 type="tns:identifier" maxOccurs="unbounded"/>
5186                         </sequence>
5187                     </complexType>
5188                 </element>
5189                 <element ref="tns:qualifiers" minOccurs="0"/>
5190             </sequence>
5191             <attribute name="id" type="ID"/>
5192             <attribute name="report-hazard" type="boolean"/>
5193         </complexType>
5194     </element>
5195
5196     <element name="request-statuses">
5197         <complexType>
5198             <sequence>
5199                 <element name="target-additional-information"
5200 type="string"/>
5201                 <element name="reply-address" type="tns:address"/>
5202                 <element name="transaction-identifier"
5203 type="tns:identifier"/>
5204                 <element name="inferiors-list" minOccurs="0">
5205                     <complexType>
5206                         <sequence>
5207                             <element name="inferior-handle"
5208 type="tns:identifier" maxOccurs="unbounded"/>
5209                         </sequence>
5210                     </complexType>
5211                 </element>

```

```

5212         <element ref="tns:qualifiers" minOccurs="0"/>
5213     </sequence>
5214     <attribute name="id" type="ID"/>
5215 </complexType>
5216 </element>
5217
5218 <element name="inferior-statuses">
5219     <complexType>
5220         <sequence>
5221             <element name="target-additional-information"
5222 type="string"/>
5223             <element name="decider-address" type="tns:address"/>
5224             <element name="transaction-identifier"
5225 type="tns:identifier"/>
5226             <element name="status-list">
5227                 <complexType>
5228                     <sequence>
5229                         <element name="status-item" minOccurs="unbounded">
5230                             <complexType>
5231                                 <sequence>
5232                                     <element name="inferior-handle"
5233 type="tns:identifier"/>
5234                                     <element name="status">
5235                                         <simpleType>
5236                                             <restriction base="string">
5237                                                 <enumeration value="active"/>
5238                                                 <enumeration value="resigned"/>
5239                                                 <enumeration value="preparing"/>
5240                                                 <enumeration value="prepared"/>
5241                                                 <enumeration value="autonomously-confirmed"/>
5242                                                 <enumeration value="autonomously-cancelled"/>
5243                                                 <enumeration value="confirming"/>
5244                                                 <enumeration value="confirmed"/>
5245                                                 <enumeration value="cancelling"/>
5246                                                 <enumeration value="cancelled"/>
5247                                                 <enumeration value="cancel-contradiction"/>
5248                                                 <enumeration value="confirm-contradiction"/>
5249                                                 <enumeration value="hazard"/>
5250                                             </restriction>
5251                                         </simpleType>
5252                                     </element>
5253                                     <element ref="tns:qualifiers" minOccurs="0"/>
5254                                 </sequence>
5255                             </complexType>
5256                         </element>
5257                     </sequence>
5258                 </complexType>
5259             </element>
5260             <element ref="tns:qualifiers" minOccurs="0"/>
5261         </sequence>
5262     <attribute name="id" type="ID"/>
5263 </complexType>
5264 </element>

```

```

5265
5266     <element name="request-status">
5267         <complexType>
5268             <sequence>
5269                 <element name="target-additional-information"
5270 type="string"/>
5271                 <element name="reply-address" type="tns:address"/>
5272                 <element name="inferior-identifier" type="tns:identifier"
5273 minOccurs="0"/>
5274                 <element name="transaction-identifier"
5275 type="tns:identifier" minOccurs="0"/>
5276                 <element ref="tns:qualifiers" minOccurs="0"/>
5277             </sequence>
5278             <attribute name="id" type="ID"/>
5279         </complexType>
5280     </element>
5281
5282     <element name="status">
5283         <complexType>
5284             <sequence>
5285                 <element name="target-additional-information"
5286 type="string"/>
5287                 <element name="inferior-address" type="tns:address"
5288 minOccurs="0"/>
5289                 <element name="inferior-identifier" type="tns:identifier"
5290 minOccurs="0"/>
5291                 <element name="decider-address" type="tns:address"
5292 minOccurs="0"/>
5293                 <element name="transaction-identifier"
5294 type="tns:identifier" minOccurs="0"/>
5295                 <element name="status-value">
5296                     <simpleType>
5297                         <restriction base="string">
5298                             <enumeration value="created"/>
5299                             <enumeration value="enrolling"/>
5300                             <enumeration value="active"/>
5301                             <enumeration value="resigning"/>
5302                             <enumeration value="resigned"/>
5303                             <enumeration value="preparing"/>
5304                             <enumeration value="prepared"/>
5305                             <enumeration value="confirming"/>
5306                             <enumeration value="confirmed"/>
5307                             <enumeration value="cancelling"/>
5308                             <enumeration value="cancelled"/>
5309                             <enumeration value="cancel-contradiction"/>
5310                             <enumeration value="confirm-contradiction"/>
5311                             <enumeration value="hazard"/>
5312                             <enumeration value="contradicted"/>
5313                             <enumeration value="unknown"/>
5314                             <enumeration value="inaccessible"/>
5315                         </restriction>
5316                     </simpleType>
5317                 </element>

```

```

5318         <element ref="tns:qualifiers" minOccurs="0"/>
5319     </sequence>
5320     <attribute name="id" type="ID"/>
5321 </complexType>
5322 </element>
5323
5324     <element name="redirect">
5325         <complexType>
5326             <sequence>
5327                 <element name="target-additional-information"
5328 type="string"/>
5329                 <element name="superior-identifier" type="tns:identifier"
5330 minOccurs="0"/>
5331                 <element name="inferior-identifier" type="tns:identifier"/>
5332                 <element name="old-address" type="tns:address"
5333 maxOccurs="unbounded"/>
5334                 <element name="new-address" type="tns:address"
5335 maxOccurs="unbounded"/>
5336                 <element ref="tns:qualifiers" minOccurs="0"/>
5337             </sequence>
5338             <attribute name="id" type="ID"/>
5339         </complexType>
5340     </element>
5341
5342     <element name="fault">
5343         <complexType>
5344             <sequence>
5345                 <element name="target-additional-information"
5346 type="string"/>
5347                 <element name="superior-identifier" type="tns:identifier"
5348 minOccurs="0"/>
5349                 <element name="inferior-identifier" type="tns:identifier"
5350 minOccurs="0"/>
5351                 <element name="fault-type" type="string"/>
5352                 <element name="fault-data" type="anyType" minOccurs="0"/>
5353                 <element ref="tns:qualifiers" minOccurs="0"/>
5354             </sequence>
5355             <attribute name="id" type="ID"/>
5356         </complexType>
5357     </element>
5358
5359 </schema>
5360

```

5360
5361

5362 **Conformance**

5363
5364
5365
5366
5367
5368

A BTP implementation need not implement all aspects of the protocol to be useful. The level of conformance of an implementation is defined by which roles it can support using the specified messages and carrier protocol bindings for interoperation with other implementations.

5369
5370
5371
5372
5373

A partially conformant implementation may implement some roles in a non-interoperable way, giving that implementation's users comparable proprietary functionality.

The following Roles and Role Groups are used to define conformance:

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

5374
5375
5376
5377
5378

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

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

5379
5380
5381
5382
5383
5384
5385
5386
5387
5388

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

5388 **Part 3. Appendices**

5389

5390

5391

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

5392

5393

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 <i>carrier protocol address</i> and an optional opaque suffix. <div style="border: 1px solid black; padding: 2px; margin-top: 5px;"><i>PRF - suffix ? I've used "additional information"</i></div>
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.
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 appropriate application operation. 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]
	<i>PRF - Sentence about countereffect contract doesn't fit well</i>
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 <i>or</i> 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.

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

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.

5394