

2 Business Transaction Protocol

3 An OASIS Committee Specification

4 ***CURRENT STATUS : internal committee draft***

5 Version 1.0 [0.9.2]

6 DD Mmm 2002 [13 February 2002 18:02]

7

8

9

10

<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
<i>Working Draft 0.9.1 – includes all issues agreed 16 Jan 2002, and 82 (deferred)</i>	18 January 2002
<i>Working Draft 0.9.1.1 – format changes and proposed soln 77,78, 17.</i>	27 January 2002
<i>Working Draft 0.9.1.2 – xml changes, new schema, and issue 74</i>	30 January 2002
<i>Working Draft 0.9.1.3 – corrections, issue 30, state table – 81, 104</i>	8 February 2002
<i>Working Draft 0.9.2 – all issues as agreed 13 February 2002</i>	13 February 2002

11 [*Change marks relative to 0.9.2*](#)

12

13

13 Copyright and related notices

14
15 Copyright © The Organization for the Advancement of Structured Information Standards
16 (OASIS), 2001. All Rights Reserved.

17
18 This document and translations of it may be copied and furnished to others, and derivative
19 works that comment on or otherwise explain it or assist in its implementation may be
20 prepared, copied, published and distributed, in whole or in part, without restriction of any
21 kind, provided that the above copyright notice and this paragraph are included on all such
22 copies and derivative works. However, this document itself may not be modified in any way,
23 such as by removing the copyright notice or references to OASIS, except as needed for the
24 purpose of developing OASIS specifications, in which case the procedures for copyrights
25 defined in the OASIS Intellectual Property Rights document must be followed, or as required
26 to translate it into languages other than English.

27
28 The limited permissions granted above are perpetual and will not be revoked by OASIS or its
29 successors or assigns.

30
31 This document and the information contained herein is provided on an "AS IS" basis and
32 OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT
33 NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION
34 HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF
35 MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

36
37
38 OASIS takes no position regarding the validity or scope of any intellectual property or other
39 rights that might be claimed to pertain to the implementation or use of the technology
40 described in this document or the extent to which any license under such rights might or
41 might not be available; neither does it represent that it has made any effort to identify any
42 such rights. Information on OASIS's procedures with respect to rights in OASIS
43 specifications can be found at the OASIS website. Copies of claims of rights made available
44 for publication and any assurances of licenses to be made available, or the result of an attempt
45 made to obtain a general license or permission for the use of such proprietary rights by
46 implementors or users of this specification, can be obtained from the OASIS Executive
47 Director.

48
49 OASIS invites any interested party to bring to its attention any copyrights, patents or patent
50 applications, or other proprietary rights which may cover technology that may be required to
51 implement this specification. Please address the information to the OASIS Executive
52 Director.

53

53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98

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.

98 **Typographical and Linguistic Conventions and Style**

99

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

103

104

Cancel

105

Participant

106

Application Message

107

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

109

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

BEGIN

118

CONTEXT

119

RESIGN

120

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

122

123

BEGIN/atom

124

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

126

127

BEGIN/atom & CONTEXT

128

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

130

131

ENROL + VOTE

132

133 XML schemata and instances are given in Courier:

134

135

```
<ctp:begin> ... </ctp:begin>
```

136

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

138

139

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

140

```
{
```

141

```
}
```

142

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

146
147
148
149
150

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

150	Contents	
151		
152	Copyright and related notices.....	2
153	Acknowledgements	3
154	Typographical and Linguistic Conventions and Style	4
155	Contents	6
156	Part 1. Purpose and Features of BTP	10
157	Introduction.....	10
158	Development and Maintenance of the Specification.....	11
159	Overview of the Business Transaction Protocol	12
160	Part 2. Normative Specification of BTP	15
161	Actors, Roles and Relationships	15
162	Relationships.....	15
163	Roles involved in the outcome relationships	17
164	Superior.....	17
165	Inferior	18
166	Enroller	19
167	Participant	20
168	Sub-coordinator.....	20
169	Sub-composer	21
170	Roles involved in the control relationships.....	21
171	Decider.....	21
172	Coordinator	22
173	Composer	22
174	Terminator.....	22
175	Initiator.....	23
176	Factory	24
177	Other roles	24
178	Redirector.....	24
179	Status Requestor.....	25
180	Abstract Messages and Associated Contracts	25
181	Addresses.....	26
182	Request/response pairs.....	27
183	Compounding messages	27
184	Extensibility.....	29
185	Messages.....	29
186	Qualifiers	29
187	Messages not restricted to outcome or control relationships.	30
188	CONTEXT.....	31
189	CONTEXT_REPLY	31
190	REQUEST_STATUS	33
191	STATUS	33
192	FAULT.....	35
193	REQUEST_INFERIOR_STATUSES, INFERIOR_STATUSES	38
194	Messages used in the outcome relationships	38
195	ENROL	38

196	ENROLLED	39
197	RESIGN	40
198	RESIGNED	41
199	PREPARE	41
200	PREPARED	42
201	CONFIRM	43
202	CONFIRMED	44
203	CANCEL	45
204	CANCELLED	46
205	CONFIRM_ONE_PHASE	47
206	HAZARD	48
207	CONTRADICTION	49
208	SUPERIOR_STATE	49
209	INFERIOR_STATE	51
210	REDIRECT	53
211	Messages used in control relationships	54
212	BEGIN	54
213	BEGUN	55
214	PREPARE_INFERIORS	56
215	CONFIRM_TRANSACTION	57
216	TRANSACTION_CONFIRMED	59
217	CANCEL_TRANSACTION	60
218	CANCEL_INFERIORS	61
219	TRANSACTION_CANCELLED	62
220	REQUEST_INFERIOR_STATUSES	62
221	INFERIOR_STATUSES	64
222	Groups – combinations of related messages	66
223	CONTEXT & application message	66
224	CONTEXT_REPLY & ENROL	67
225	CONTEXT_REPLY (& ENROL) & PREPARED / & CANCELLED	68
226	CONTEXT_REPLY & ENROL & application message (& PREPARED)	68
227	BEGUN & CONTEXT	69
228	BEGIN & CONTEXT	69
229	Standard qualifiers	69
230	Transaction timelimit	69
231	Inferior timeout	70
232	Minimum inferior timeout	71
233	Inferior name	72
234	State Tables	73
235	Explanation of the state tables	73
236	Status queries	73
237	Decision events	73
238	Disruptions – failure events	74
239	Invalid cells and assumptions of the communication mechanism	74
240	Meaning of state table events	75
241	Persistent information	79
242	Failure Recovery	92

243	Types of failure	92
244	Persistent information	93
245	Redirection.....	94
246	Terminator:Decider failures.....	95
247	XML representation of Message Set.....	95
248	Addresses	96
249	Qualifiers	96
250	Identifiers	97
251	Message References.....	97
252	Messages.....	97
253	CONTEXT.....	97
254	CONTEXT_REPLY	97
255	REQUEST_STATUS	98
256	STATUS	98
257	FAULT.....	98
258	ENROL	99
259	ENROLLED	100
260	RESIGN	100
261	RESIGNED.....	100
262	PREPARE	101
263	PREPARED	101
264	CONFIRM	101
265	CONFIRMED	101
266	CANCEL	102
267	CANCELLED.....	102
268	CONFIRM_ONE_PHASE	102
269	HAZARD.....	103
270	CONTRADICTION.....	103
271	SUPERIOR_STATE.....	103
272	INFERIOR_STATE.....	103
273	REDIRECT.....	104
274	BEGIN	104
275	BEGUN.....	104
276	PREPARE_INFERIORS	105
277	CONFIRM_TRANSACTION	105
278	TRANSACTION_CONFIRMED.....	106
279	CANCEL_TRANSACTION	106
280	CANCEL_INFERIORS	106
281	TRANSACTION_CANCELLED.....	107
282	REQUEST_INFERIOR_STATUSES	107
283	INFERIOR_STATUSES	107
284	Standard qualifiers	108
285	Transaction timelimit.....	108
286	Inferior timeout	108
287	Minimum inferior timeout	108
288	Inferior name.....	108
289	Compounding of Messages.....	108

290	XML Schemas	110
291	XML schema for BTP messages.....	110
292	XML schema for standard qualifiers	123
293	Carrier Protocol Bindings	125
294	Carrier Protocol Binding Proforma.....	125
295	Bindings for request/response carrier protocols	126
296	Request/response exploitation rules.....	127
297	SOAP Binding	128
298	Example scenario using SOAP binding	130
299	SOAP + Attachments Binding.....	132
300	Conformance	134
301	Part 3. Appendices.....	137
302	A. Glossary.....	137
303		
304		

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.

347 **Development and Maintenance of the Specification**

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

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

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

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

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

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

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

376
377 The main list of the committee is:

378
379 business-transaction@lists.oasis-open.org
380

381
382
383
384
385

385 Overview of the Business Transaction Protocol

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

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

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

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

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

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

415
416
417
418 After receiving these reports, the coordinating entity is responsible for determining which of
419 the parties should be instructed to confirm and which should be instructed to cancel.

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

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

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

434

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

446

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

455

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

467

468 It should be noted that this BTP specification recognises that:

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

479 triggered by a cancel order); or by remembering the operations (having checked they
480 are valid) and performing them only if a confirm order is received; or by forbidding
481 any other access to data changed by the operations and releasing them in their
482 unchanged state (if cancelled) or their changed state (if confirmed); or by various
483 combinations of these. In addition, a cancellation may not return data to their original
484 state, but only to a state accepted by the application as appropriate to a cancelled
485 operation.
486
487
488
489
490
491
492

Part 2. Normative Specification of BTP

Actors, Roles and Relationships

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

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

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

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

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

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

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

Relationships

There are two primary relationships in BTP.

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

535

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

538

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

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

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

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

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

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

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

553 7. Eventually agreement (or not) is reported to the Decider. If all have agreed, the
554 Decider makes and persists the confirm decision (hence the term “Decider” – it
555 decides, everything else just asked); if any have disagreed, or if the confirm decision
556 cannot be persisted, a cancel decision is made

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

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

559 10. The Decider replies to the Terminator’s request to confirm, reporting the outcome
560 decision

561

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

567

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

571

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

575

- 576 2. The Superior:Inferior relationship is recoverable – depending on the progress of the
577 relationship, the two sides will re-establish their shared state after failure; the
578 Terminator:Decider relationship is not recoverable
579
- 580 3. The nature of the Superior:Inferior relationship requires that the two parties know of
581 each other’s addresses from when the relationship is established; the Decider does not
582 need to know the address of the Terminator (provided it has some way of returning
583 the response to a received message).
584

585 In the following sections, the responsibility of each role is defined, and the messages that are
586 sent or received by that role are listed. Note that some roles exist only to have a name for an
587 actor that issues a message and receives a reply to that message. Some of these roles may be
588 played by several actors in the course of a single business transaction.
589

590 **Roles involved in the outcome relationships**

591

592 **Superior**

593

594 Accepts enrolments from Inferiors, establishing a Superior:Inferior relationship with each. In
595 cooperation with other actors and constrained by the messages exchanged with the Inferior,
596 the Superior determines the **Outcome** applicable to the Inferior and informs the Inferior by
597 sending CONFIRM or CANCEL. This outcome can be confirm only if a PREPARED
598 message is received from the Inferior, and if a record, identifying the Inferior can be
599 persisted. (Whether this record is also a record of a confirm decision depends on the
600 Superior’s position in the business transaction as a whole.). The Superior must retain this
601 persistent record until it receives a CONFIRMED (or, in exceptional cases, CANCELLED or
602 HAZARD) from the Inferior.
603

604

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

607

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

612

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

615

616 A Superior receives

617

618 ENROL

619

620 to enrol a new Inferior, establishing a new Superior:Inferior relationship.

621

622 A Superior sends

623

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

626
627 A Superior sends

628
629 PREPARE
630 CONFIRM
631 CANCEL
632 RESIGNED
633 CONFIRM_ONE_PHASE
634 SUPERIOR_STATE

635
636 to an enrolled Inferior.

637
638 A Superior receives

639
640 PREPARED
641 CANCELLED
642 CONFIRMED
643 HAZARD
644 RESIGN
645 INFERIOR_STATE

646
647 from an enrolled Inferior.

648
649 **Inferior**

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

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

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

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

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

673

674 An Inferior receives

675

676 PREPARE
677 CONFIRM
678 CANCEL
679 RESIGNED
680 CONFIRM_ONE_PHASE
681 SUPERIOR_STATE

682

683 from its Superior.

684

685 An Inferior sends

686

687 PREPARED
688 CANCELLED
689 CONFIRMED
690 HAZARD
691 RESIGN
692 INFERIOR_STATE

693

694 to its Superior.

695

696

697 **Enroller**

698

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

704

705 An Enroller sends

706

707 ENROL

708

709 to a Superior.

710

711 An Enroller receives

712

713 ENROLLED

714

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

716

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

723 Participant

724
725 An Inferior which is specialized for the purposes of an application. Some application
726 operations are associated directly with the Participant, which is responsible for determining
727 whether a prepared condition is possible for them, and for applying the outcome. (“associated
728 directly” as opposed to involving another BTP Superior:Inferior relationship, in which this
729 actor is the Superior).

730
731 The associated operations may be performed by the actor that has the role of Participant, or
732 they may be performed by another actor, and only the confirm/cancel application is
733 performed by the Participant.
734

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

739 Note – Possible approaches are:

- 740 o The operations may be performed completely and the
741 Participant persists information to perform counter-effect
742 operations (compensating operations) to apply
743 cancellation;
 - 744 o The operations may be just checked and not performed at
745 all; the Participant persists information to perform them to
746 apply confirmation;
 - 747 o The Participants persists the prior state of data affected by
748 the operations and the operations are performed; the
749 Participant restores the prior state to apply cancellation;
 - 750 o As the previous, but other access to the affected data is
751 forbidden until the decision is known
-

752 Sub-coordinator

753
754 An Inferior which is also an Atomic Superior.
755

756
757 A sub-coordinator is the Inferior in one Superior:Inferior relationship and the Superior in one
758 or more Superior:Inferior relationships.

759
760 From the perspective of its Superior (the one the sub-coordinator is Inferior to), there is no
761 difference between a sub-coordinator and any other Inferior. From this perspective, the
762 “associated operations” of the sub-coordinator as an Inferior include the relationships with its
763 Inferiors.

764
765 A sub-coordinator does not become prepared (and send PREPARED to its Superior) until and
766 unless it has received PREPARED (or RESIGN) from all its Inferiors. The outcome is
767 propagated to all Inferiors.

768 **Sub-composer**

769
770 An Inferior which is also a Cohesive Superior.

771
772
773 Like a sub-coordinator, a sub-composer cannot be distinguished from any other Inferior from
774 the perspective of its Superior.

775
776 A sub-composer is similar to a sub-coordinator, except that the constraints linking the
777 different Inferiors concern only those Inferiors in the confirm-set. How the confirm-set is
778 controlled, and when, is not defined in this specification.

779
780 If the sub-composer is instructed to cancel, by receiving a CANCEL message from its
781 Superior, the cancellation is propagated to all its Inferiors.

782
783

784 **Roles involved in the control relationships**

785

786 **Decider**

787

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

794

795 A Decider is instructed to cancel by receiving CANCEL_TRANSACTION.

796

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

800

801 All Deciders receive

802 CONFIRM_TRANSACTION

803 CANCEL_TRANSACTION

804 REQUEST_INFERIOR_STATUSES

805

806 All Deciders send
807 CONFIRM_COMPLETE
808 CANCEL_COMPLETE
809 INFERIOR_STATUSES

810
811

812 Coordinator

813

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

816

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

818

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

823

824 Composer

825

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

828

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

831

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

836

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

841

842 A Composer may be asked to cancel some of its Inferiors, but not itself, by receiving
843 CANCEL_INFERIORS.

844

845

846 Terminator

847

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

850

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

853
854 A request to confirm is made by sending CONFIRM_TRANSACTION to the target Decider.
855 If the Decider is a Cohesion Composer, the Terminator may select which of the Composer's
856 Inferiors are to be included in the confirm-set. If the Decider is an Atom Coordinator, all
857 Inferiors are included. After applying the decision, the Decider replies with
858 CONFIRM_COMPLETE, CANCEL_COMPLETE or (in the case of problems)
859 INFERIOR_STATUSES.

860
861 A Terminator may ask a Composer (but not a Coordinator) to prepare some or all of its
862 Inferiors with PREPARE_INFERIORS. The Composer replies with
863 INFERIOR_STATUSES.

864
865 A Terminator may send CANCEL_TRANSACTION to instruct the Decider to cancel the
866 whole business transaction.,. The Decider replies with CANCEL_COMPLETE if all Inferiors
867 cancel successfully, and with INFERIOR_STATUSES in the case of problems.. If the
868 Decider is a Cohesion Composer, the Terminator may send CANCEL_INFERIORS to cancel
869 some of the Inferiors; the Decider always replies with INFERIOR_STATUSES.

870
871 A Terminator may check the status of the Inferiors of the Decider by sending
872 REQUEST_INFERIOR_STATUSES. The Decider replies with INFERIOR_STATUSES.

873
874 A Terminator sends
875 CONFIRM_TRANSACTION
876 CANCEL_TRANSACTION
877 CANCEL_INFERIORS
878 PREPARE_INFERIORS
879 REQUEST_INFERIOR_STATUSES

880
881 A Terminator receives
882 CONFIRM_COMPLETE
883 CANCEL_COMPLETE
884 INFERIOR_STATUSES

885 886 Initiator

887
888 Requests a **Factory** to create a Superior – this will either be a Decider (representing a new
889 top-level business transaction) or a sub-coordinator or sub-composer to be the Inferior of an
890 existing business transaction.

891 An Initiator sends

892 BEGIN
893 BEGIN & CONTEXT

894 to a Factory, and receives in reply

895 BEGUN & CONTEXT
896

900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946

Factory

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

Decider, which is either
Composer or
Coordinator
Sub-composer
Sub-coordinator

A Factory receives

BEGIN
BEGIN & CONTEXT

and replies with

BEGUN & CONTEXT

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

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

Other roles

Redirector

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

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

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

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

948

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

951

952 **Status Requestor**

953

954 Requests and receives the current status of a transaction tree node – any of an Inferior,
955 Superior or Decider, or the current status of the nodes relationships with its Inferiors, if any.
956 The role of Status Requestor has no responsibilities – it is just a name for where the
957 REQUEST_STATUS and REQUEST_INFERIOR_STATUSES comes from
958 (REQUEST_INFERIOR_STATUSES is also issued by a Terminator to a Decider).

959

960 A Status Requestor sends

961

962 REQUEST_STATUS

963 REQUEST_INFERIOR_STATUSES

964

965 and receives

966

967 STATUS

968 INFERIOR_STATUSES

969

970 in response.

971

972 The receiver of the request can refuse to provide the status information by replying with
973 FAULT(StatusRefused). The information returned in STATUS will always relate to the
974 transaction tree node as a whole (e.g. as an Inferior, even if it is also a Superior).

975

976 **Abstract Messages and Associated Contracts**

977

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

981

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

983

984 deliver messages completely and correctly, or not at all (corrupted messages will
985 not be delivered);

986

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

989

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

991

992 and

993

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

995

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

998

999 However, when the abstract message set is mapped to a carrier protocol that provides a richer
1000 service (e.g. reports all delivery failures, guarantees ordered delivery or offers a
1001 request/response mechanism), the mapping can take advantage of these features. Typically in
1002 such cases, some of the parameters of an abstract message will be implicit in the carrier
1003 mechanisms, while the values of other parameters will be directly represented in transmitted
1004 elements.

1005

1006

1007 **Addresses**

1008

1009 All of the messages except CONTEXT have a “target address” parameter and many also have
1010 other address parameters. These latter identify the desired target of other messages in the set.
1011 In all cases, the exact value will invariably have been originally determined by the
1012 implementation that is the target or desired future target.

1013

1014 The detailed format of the address will depend on the particular carrier protocol, but at this
1015 abstract level is considered to have three parts. The first part, the “binding name”, identifies
1016 the binding to a particular carrier protocol – some bindings are specified in this document,
1017 others can be specified elsewhere. The second part of the address, the “binding address”, is
1018 meaningful to the carrier protocol itself, which will use it for the communication (i.e. it will
1019 permit a message to be delivered to a receiver). The third part, “additional information”, is
1020 not used or understood by the carrier protocol. The “additional information” may be a
1021 structured value.

1022

1023 When a message is actually transmitted, the “binding name” of the target address will identify
1024 which carrier protocol is in use and the “binding address” will identify the destination, as
1025 known to the carrier protocol. The entire binding address is considered to be “consumed” by
1026 the carrier protocol implementation. All of it may be used by the sending implementation, or
1027 some of it may be transmitted in headers, or as part of a URL in the carrier protocol, but then
1028 used or consumed by the receiving implementation of the carrier protocol to direct the BTP
1029 message to a BTP-aware entity (BTP-aware in that it is capable of interpreting the BTP
1030 messages). The “additional information” of the target address will be part of the BTP
1031 message itself and used in some way by the receiving BTP-aware entity (it could be used to
1032 route the message on to some other BTP entity). Thus, for the target address, only the
1033 “additional information” field is transmitted in the BTP message and the “additional
1034 information” is opaque to parties other than the recipient.

1035

1036 For other addresses in BTP messages, all three components will be within the message.

1037

1038 All messages that concern a particular Superior:Inferior relationship have an identifier
1039 parameter for the target side as well as the target address. This allows full flexibility for
1040 implementation choices – an implementation can:

- 1041
1042 a) Use the same binding address and additional information for multiple business
1043 transactions, using the identifier parameter to locate the relevant state
1044 information;
1045 b) Use the same binding address for multiple business transactions and use the
1046 additional information to locate the information; or
1047 c) Use a different binding address for each business transaction.
1048

1049 Which of these choices is used is opaque to the entity sending the message – both parts of the
1050 address and the identifier originated at the recipient of this message (and were transmitted as
1051 parameters of earlier messages in the opposite direction).
1052

1053 BTP recovery requires that the state information for a Superior or Inferior is accessible after
1054 failure and that the peer can distinguish between temporary inaccessibility and the permanent
1055 non-existence of the state information. As is explained in “Redirection” below, BTP provides
1056 mechanisms – having a set of BTP addresses for some parameters, and the REDIRECT
1057 message – that make this possible, even if the recovered state information is on a different
1058 address to the original one (as may be the case if case c) above is used).
1059
1060

1061 **Request/response pairs**

1062

1063 Many of the messages combine in pairs as a request and its response. However, in some cases
1064 the response message is sent without a triggering request, or as a possible response to more
1065 than one type of request. To allow for this, the abstract message set treats each message as
1066 standalone; but where a request does expect a reply, a “reply-address” parameter will be
1067 present. For any message with a reply address parameter, in the case of certain errors, a
1068 FAULT message will be sent to the reply address instead of the expected reply.
1069

1070 For messages which are specified as sent between Superior and Inferior, a FAULT message is
1071 sent to the peer.
1072

1073 **Compounding messages**

1074

1075 BTP messages may be sent in combination with each other, or with other (application)
1076 messages. There are two cases:
1077

- 1078 a) Sending the messages together where the combination has semantic
1079 significance. One message is said to be “related to” the other – the combination
1080 is termed a “group”.
1081 b) Sending of the messages where the combination has no semantic significance,
1082 but is merely a convenience or optimisation. This is termed “bundling” – the
1083 combination is termed a “bundle”.
1084

1085 The form A&B is used to refer to a combination (group) where message B is sent in relation
1086 to A (“relation” is asymmetric). The form A+B is used to refer to A and B bundled together-

1087 the transmission of the bundle "A+B" is semantically identical to the transmission of A
1088 followed by the transmission of B.

1089
1090 Only certain combinations of messages are possible in a group, and the meaning of the
1091 relation is specifically defined for each such combination in the next section. A particular
1092 group is treated as a unit for transmission – it has a single target address. This is usually that
1093 of one of the messages in the group – the specification for the group defines which.

1094
1095 A “bundle” of messages may contain both unrelated messages and groups of related
1096 messages. The only constraint on which messages and groups can be bundled is that all have
1097 the same binding address, but may have different “additional information” values. (Messages
1098 within a related group may have different addresses, where the rules of their relatedness
1099 permit this). Unless constrained by the binding, any messages or groups that are to be sent to
1100 the same binding address may be bundled – the fact that the binding addresses are the same is
1101 a necessary and sufficient condition for the sender to determine that the messages can be
1102 bundled.

1103
1104 A particular and important case of related messages is where a BTP CONTEXT message is
1105 sent related to an application message. In this case, the target of the application message
1106 defines the destination of the CONTEXT message. The receiving implementation may in fact
1107 remove the CONTEXT before delivering the application message to the application (Service)
1108 proper, but from the perspective of the sender, the two are sent to the same place.
1109 The compounding mechanisms, and the multi-part address structures, support the “one-wire”
1110 and “one-shot” communication patterns.

1111
1112 In “one-wire”, all message exchanges between two sides of a Superior:Inferior relationship,
1113 including the associated application messages, pass via the same “endpoints”. These
1114 “endpoints” may in fact be relays, routing messages on to particular actors within their
1115 domain. The onward routing will require some further addressing, but this has to be opaque to
1116 the sender. This can be achieved if the relaying endpoint ensures that all addresses for actors
1117 in its domain have the relay’s address as their binding address, and any routing information it
1118 will need in its own domain is placed in the additional information. (This may involve the
1119 relay changing addresses in messages as they pass through it on the way out). On receiving a
1120 message, it determines the within-domain destination from the received additional
1121 information (which is thus rewritten) and forwards the message appropriately. The sender is
1122 unaware of this, and merely sees addresses with the same binding address, which it is
1123 permitted to bundle. The content of the “additional information” is a matter only for the relay
1124 – it could put an entire BTP address in there, or other implementation-defined information.
1125 Note that a quite different one-wire implementation can be constructed where there is no
1126 relaying, but the receiving entity effectively performs all roles, using the received identifiers
1127 to locate the appropriate state.

1128
1129 “One-shot” communication makes it possible to send an application message, receive the
1130 application reply, enrol an Inferior to be responsible for the confirm/cancel of the operations
1131 of those message and inform the Superior that the Inferior is prepared, all in one two-way
1132 exchange across the network (e.g. one request/reply of a carrier protocol).. The application
1133 request is sent with a related CONTEXT message. The application response is sent with a

1134 relation group of CONTEXT_REPLY/related, ENROL/no-rsp-req message and a
1135 PREPARED message. This is possible even if the Superior address is different from the
1136 address of the application element that sends the original message (if the application
1137 exchange is request/reply, there may not even be an identifiable address for the application
1138 element). The target addresses of the ENROL and PREPARED (the Superior address) are not
1139 transmitted; the actor that was originally responsible for adding the CONTEXT to the
1140 outbound application message remembers the Superior address and forwards the ENROL and
1141 PREPARED appropriately.

1142
1143 With “one-shot”, if there are multiple Inferiors created as a result of a single application
1144 message, there is an ENROL and PREPARED message for each sent related to the
1145 CONTEXT_REPLY. If an operation fails, a CANCELLED message is sent instead of a
1146 PREPARED.

1147
1148 If the CONTEXT has “superior-type” of “atom”, then subsequent messages to the same
1149 Service, with the same related CONTEXT/atom, can have their associated operations put
1150 under the control of the same Inferior, and only a CONTEXT_REPLY/completed is sent back
1151 with the response (if the new operations fail, it will be necessary to send back
1152 CONTEXT_REPLY/repudiated, or send CANCELLED). If the “superior type” on the
1153 CONTEXT is “cohesive”, each operation will require separate enrolment.

1154
1155 Whether the “one-shot” mechanism is used is determined by the implementation on the
1156 responding (Inferior) side. This may be subject to configuration and may also be constrained
1157 by the application or by the binding in use.

1158

1159 **Extensibility**

1160

1161 To simplify interoperation between implementations of this edition of BTP with
1162 implementations of future editions, the “must-be-understood” sub-parameter as specified for
1163 Qualifiers may be defined for use with any parameter added to an existing message in a future
1164 revision of this specification. The default for “must-be-understood” shall be “true”, so an
1165 implementation receiving an unrecognised parameter without a “false” value for “must-be-
1166 understood” shall not accept it (the FAULT value “UnrecognisedParameter” is available, but
1167 other errors, including lower-layer parsing/unmarshalling errors may be reported instead). If
1168 “must-be-understood” with the value “false” is present as a sub-parameter of a parameter in
1169 any message, a receiving implementation **should** ignore the parameter.

1170

1171 How the sub-parameter is associated with the new parameter is determined by the particular
1172 binding.

1173

1174 No special mechanism is provided to allow for the introduction of completely new messages.

1175

1176 **Messages**

1177

1178 **Qualifiers**

1179

1180 All messages have a Qualifiers parameter which contains zero or more Qualifier values. A
1181 Qualifier has sub-parameters:
1182

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

1183
1184 **Qualifier group** ensures the Qualifier name is unambiguous. Qualifiers in the
1185 same group need not have any functional relationship. The qualifier group will
1186 typically be used to identify the specification that defines the qualifier’s meaning
1187 and use. Qualifiers may be defined in this or other standard specifications, in
1188 specifications of a particular community of users or of implementations or by
1189 bilateral agreement.

1190
1191 **Qualifier name** this identifies the meaning and use of the Qualifier, using a name
1192 that is unambiguous within the scope of the Qualifier group.

1193
1194 **Must-be-understood** if this has the value “true” and the receiving entity does
1195 not recognise the Qualifier type (or does not implement the necessary
1196 functionality), a FAULT “UnsupportedQualifier” shall be returned and the
1197 message shall not be processed. Default is “true”.

1198
1199 **To-be-propagated** if this has the value “true” and the receiving entity passes the
1200 BTP message (which may be a CONTEXT, but can be other messages) onwards
1201 to other entities, the same Qualifier value shall be included. If the value is
1202 “false”, the Qualifier shall not be automatically included if the BTP message is
1203 passed onwards. (If the receiving entity does support the qualifier type, it is
1204 possible a propagated message may contain another instance of the same type,
1205 even with the same Content – this is not considered propagation of the original
1206 qualifier.). Default is “false”.

1207
1208 **Content** the type (which may be structured) and meaning of the content is
1209 defined by the specification of the Qualifier.

1210 **Messages not restricted to outcome or control relationships.**

1211
1212
1213
1214 The messages in this section are used between various roles. CONTEXT message is used in
1215 the Initiator:Factory relationship (when it is related to BEGIN or to BEGUN), and related to
1216 an application ‘message’ to propagate the business transaction between parts of the
1217 application. CONTEXT_REPLY is used as the reply to a CONTEXT.REQUEST_STATUS

1218 can be issued to, and STATUS returned by any of Decider, Superior or Inferior. FAULT can
1219 be used on any relationship to indicate an error condition back to the sender of a message.

1220

1221 CONTEXT

1222

1223 A CONTEXT is supplied by (or on behalf of) a Superior and related to one or more
1224 application messages. (The means by which this relationship is represented is determined by
1225 the binding and the binding mechanisms of the application protocol.) The “superior type”
1226 parameter identifies whether the Superior will apply the same decision to all Inferiors
1227 enrolled using the same superior identifier (“superior type” is “atom”) or whether it may
1228 apply different decisions (“superior type” is “cohesion”).

1229

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

1230

1231

1232 **address-as-superior** the address to which ENROL and other messages from an
1233 enrolled Inferior are to be sent. This can be a set of alternative addresses.

1234

1235 **superior identifier** identifies the Superior. This shall be globally unambiguous.
1236 **reply-address** the address to which a replying CONTEXT_REPLY is to be sent.
1237 This may be different each time the CONTEXT is transmitted – it refers to the
1238 destination of a replying CONTEXT_REPLY for this particular transmission of
1239 the CONTEXT.

1240

1241 **superior type** identifies whether the CONTEXT refers to a Cohesion or an
1242 Atom. Default is atom.

1243

1244 **qualifiers** standardised or other qualifiers. The standard qualifier “Transaction
1245 timelimit” is carried by CONTEXT.

1246

1247 There is no target address parameter for CONTEXT as it is only transmitted in relation to the
1248 application messages, BEGIN and BEGUN.

1249

1250 The forms CONTEXT/cohesion and CONTEXT/atom refer to CONTEXT messages with the
1251 superior type with the appropriate value.

1252

1253

1254 CONTEXT_REPLY

1255

1256 CONTEXT_REPLY is sent after receipt of CONTEXT (related to application message(s)) to
 1257 indicate whether all necessary enrolments have already completed (ENROLLED has been
 1258 received) or will be completed by ENROL messages sent in relation to the
 1259 CONTEXT_REPLY or if an enrolment attempt has failed. CONTEXT_REPLY may be sent
 1260 related to an application message (typically the response to the application message related to
 1261 the CONTEXT). In some bindings the CONTEXT_REPLY may be implicit in the application
 1262 message.
 1263

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

1264
 1265 **target-address** the address to which the CONTEXT_REPLY is sent. This shall
 1266 be the “reply-address” from the CONTEXT.
 1267

1268 **superior identifier** the superior identifier from the CONTEXT
 1269

1270 **completion_status:** reports whether all enrol operations made necessary by the
 1271 receipt of the earlier CONTEXT message have completed. Values are
 1272

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.

1273
 1274 **qualifiers** standardised or other qualifiers.
 1275

1276 The form CONTEXT_REPLY/completed, CONTEXT_REPLY/related and
 1277 CONTEXT_REPLY/repudiated refer to CONTEXT_REPLY messages with status having the
 1278 appropriate value. The form CONTEXT_REPLY/ok refers to either of
 1279 CONTEXT_REPLY/completed or CONTEXT_REPLY/related.
 1280

1281 If there are no necessary enrolments (e.g. the application messages related to the received
 1282 CONTEXT did not require the enrolment of any Inferiors), then
 1283 CONTEXT_REPLY/completed is used.
 1284

1285 If a CONTEXT_REPLY/repudiated is received, the receiving implementation **must** ensure
 1286 that the business transaction will not be confirmed.

1287
1288
1289
1290
1291
1292
1293

REQUEST_STATUS

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

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

1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308

target address the address to which the REQUEST_STATUS message is sent. This can be any of address-as-decider, address-as-inferior or address-as-superior.

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

target identifier The identifier for the business transaction, or part of business transaction whose status is sought. If the target-address is an address-as-decider, this parameter shall be the “transaction-identifier” on the BEGUN message. If the target-address is an address-as-inferior, this parameter shall be the “inferior-identifier” on the ENROL message. If the target-address is a an address-as-superior, this parameter shall be the “superior-identifier” on the CONTEXT.

qualifiers standardised or other qualifiers.

Types of FAULT possible (sent to reply address)

General

StatusRefused – if the receiver is not prepared to report its status to the sender of this message

UnknownTransaction – if the target-identifier is unknown

1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321

STATUS

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 sender.

Parameter	Type
target address	BTP address

responders-identifier	Identifier
status	See below
qualifiers	List of qualifiers

1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334

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

responders-identifier the identifier of the state, identical to the “target-identifier” on the REQUEST_STATUS.

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 Superior	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	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 or CONFIRM has been received as Inferior but responses from inferiors are pending	CONFIRM has been received; CONFIRMED/response has not been sent
<i>Confirmed</i>	CONFIRMED/responses have been received from all Inferiors	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

status value	Meaning from Superior	Meaning from Inferior
<i>Cancelled</i>	CANCELLED has been received from all Inferiors	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; 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 target-identifier exists	No state information for the target-identifier exists
<i>Inaccessible</i>	There may be state information for this target-identifier but it cannot be reached/existence cannot be determined	There may be state information for this target-identifier but it cannot be reached/existence cannot be determined

1335

1336

qualifiers standardised or other qualifiers.

1337

1338

Types of FAULT possible

1339

1340

General

1341

1342

FAULT

1343

1344

Sent in reply to various messages to report an error condition

1345

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

1346
1347 **target address** the address to which the FAULT is sent. This may be the reply
1348 address from a received message or the address of the opposite side
1349 (superior/inferior) as given in a CONTEXT or ENROL message
1350
1351 **superior identifier** the superior identifier as on the CONTEXT message and as
1352 used on the ENROL message (present only if the FAULT is sent to the superior).
1353
1354 **inferior identifier** the inferior identifier as on the ENROL message (present only
1355 if the FAULT is sent to the inferior)
1356
1357 **fault type** identifies the nature of the error, as specified for each of the main
1358 messages.
1359
1360 **fault data** information relevant to the particular error. Each fault type defines the
1361 content of the fault data:
1362

1363

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

1364

1365 *UnknownParameter* A BTP message has been Free text explanation
 1366 received with an unrecognised
 1367 **q** parameter
 1368 **u**
 1369 **Qualifiers** standardised or other qualifiers.
 1370

1371 Note – If the carrier mechanism used for the transmission of BTP messages
 1372 is capable of delivering messages in a different order than they were sent in,
 1373 the “WrongState” FAULT is not sent and should be ignored if received.

1374
 1375 **REQUEST_INFERIOR_STATUSES, INFERIOR_STATUSES**

1376
 1377 REQUEST_INFERIOR_STATUSES may be sent to and INFERIOR_STATUSES sent from
 1378 any Decider, Superior or Inferior, asking it to report on the status of its relationships with
 1379 Inferiors (if any). Since Deciders are required to respond to
 1380 REQUEST_INFERIOR_STATUSES with INFERIOR_STATUSES but non-Deciders may
 1381 just issue FAULT(StatusRefused), and INFERIOR_STATUSES is also used as a reply to
 1382 other messages from Terminator to Decider, these messages are described below under the
 1383 messages used in the control relationships.
 1384

1385 **Messages used in the outcome relationships**

1386
 1387 **ENROL**

1388
 1389 A request to a Superior to ENROL an Inferior. This is typically issued after receipt of a
 1390 CONTEXT message in relation to an application request.
 1391 The actor issuing ENROL plays the role of Enroller.
 1392

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

1393
 1394 **target address** the address to which the ENROL is sent. This will be the
 1395 address-as-superior from the CONTEXT message.
 1396

1397 **superior identifier.** The superior identifier as on the CONTEXT message
1398
1399 **reply requested** true if an ENROLLED response is required, false otherwise.
1400 Default is false.
1401
1402 **reply address** the address to which a replying ENROLLED is to be sent, if
1403 “reply requested” is true. If this field is absent and “reply requested” is true, the
1404 ENROLLED should be sent to the “address-as-inferior” (or one of them, at
1405 sender’s option)
1406
1407 **address-as-inferior** the address to which PREPARE, CONFIRM, CANCEL and
1408 SUPERIOR_STATE messages for this Inferior are to be sent.
1409
1410 **inferior identifier** an identifier that identifies this Inferior. This shall be globally
1411 unambiguous..
1412
1413 **qualifiers** standardised or other qualifiers. The standard qualifier “Inferior
1414 name” may be present.
1415

1416 Types of FAULT possible (sent to Reply address)

1417
1418 **General**

1419 **InvalidSuperior** – if superior identifier is unknown

1420 **DuplicateInferior** – if inferior with at least one of the set address-as-
1421 inferior the same and the same inferior identifier is already enrolled

1422 **WrongState** – if it is too late to enrol new Inferiors (generally if the
1423 Superior has already sent a PREPARED message to its superior or
1424 terminator, or if it has already issued CONFIRM to other Inferiors).
1425

1426 The form ENROL/rsp-req refers to an ENROL message with “reply requested” having the
1427 value “true”; ENROL/no-rsp-req refers to an ENROL message with “reply requested” having
1428 the value “false”
1429

1430 ENROL/no-rsp-req is typically sent in relation to CONTEXT_REPLY/related. ENROL/rsp-
1431 req is typically when CONTEXT_REPLY/completed will be used (after the ENROLLED
1432 message has been received.)
1433

1434 **ENROLLED**

1435
1436 Sent from Superior in reply to an ENROL/rsp-req message, to indicate the Inferior has been
1437 successfully enrolled (and will therefore be included in the termination exchanges)
1438

Parameter	Type
target address	BTP address
inferior identifier	Identifier

Parameter	Type
Qualifiers	List of qualifiers
1439	
1440	target address the address to which the ENROLLED is sent. This will be the
1441	reply address from the ENROL message (or one of the address-as-inferiors if the
1442	reply address was empty)
1443	
1444	inferior identifier The inferior identifier as on the ENROL message
1445	
1446	qualifiers standardised or other qualifiers.
1447	

1448 No FAULT messages are issued on receiving ENROLLED.

1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460

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
inferior identifier	identifier
response requested	Boolean
Qualifiers	List of qualifiers

1461

1462 **target address** the address to which the RESIGN is sent. This will be the

1463 superior address as used on the ENROL message.

1464

1465 **superior-identifier** The superior identifier as on the ENROL message

1466

1467 **inferior-identifier** The inferior identifier as on the earlier ENROL message

1468

1469 **response-requested** is set to “true” if a RESIGNED response is required.

1470

1471 **qualifiers** standardised or other qualifiers.

1472

1473 Note -- RESIGN is equivalent to readonly vote in some other protocols, but can be issued
1474 early.

1475
1476 Types of FAULT possible (sent to address-as-inferior)

1477
1478 **General**
1479 **InvalidSuperior** – if superior identifier is unknown
1480 **InvalidInferior** – if no ENROL had been received for this address-as-
1481 inferior and identifier (Inferior Identity)
1482 **WrongState** – if a PREPARED or CANCELLED has already been
1483 received by the Superior from this Inferior
1484

1485 The form RESIGN/rsp-req refers to an RESIGN message with “reply requested” having the
1486 value “true”; RESIGN /no-rsp-req refers to an RESIGN message with “reply requested”
1487 having the value “false”
1488

1489

1490 RESIGNED

1491
1492 Sent in reply to a RESIGN/rsp-req message.
1493

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

1494
1495 **target address** the address to which the RESIGNED is sent. This will be the
1496 address-as-inferior from the ENROL message.
1497

1498 **inferior identifier** The inferior identifier as on the earlier ENROL message for
1499 this Inferior.
1500

1501 **qualifiers** standardised or other qualifiers.
1502

1503 After receiving this message the Inferior will not receive any more messages with this
1504 address-as-inferior and identifier.
1505

1506 No FAULT messages are issued on receiving RESIGNED.
1507

1508 PREPARE

1509
1510 Sent from Superior to an Inferior from whom ENROL but neither CANCELLED nor
1511 RESIGN have been received, requesting a PREPARED message. PREPARE can be sent after
1512 receiving a PREPARED message.
1513

1514

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

1515

1516

1517

1518

1519

target address the address to which the PREPARE message is sent. When sent from Superior to Inferior, this will be the address-as-inferior from the ENROL message.

1520

1521

1522

inferior identifier When sent from Superior to Inferior, the inferior identifier as on the earlier ENROL message.

1523

1524

1525

1526

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

1527

1528

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

1529

1530

Types of FAULT possible (sent to Superior address)

1531

1532

General

1533

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

1534

1535

WrongState – if a CONFIRM or CANCEL has already been received by this Inferior.

1536

1537

1538

1539

PREPARED

1540

1541

1542

1543

1544

1545

1546

1547

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

Parameter	Type
target address	BTP address
superior identifier	Identifier
inferior identifier	Identifier
default is cancel	Boolean

qualifiers

List of qualifiers

1548

1549

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

1550

1551

1552

superior identifier the superior identifier as on the ENROL message

1553

1554

inferior identifier The inferior identifier as on the ENROL message

1555

1556

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.

1557

1558

1559

1560

1561

1562

1563

1564

1565

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

1566

1567

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.

1568

1569

1570

1571

1572

1573

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

1574

1575

1576

General

1577

InvalidSuperior – if Superior identifier is unknown

1578

InvalidInferior – if no ENROL has been received for this address-as-

1579

inferior and identifier, or if RESIGN has been received from this Inferior

1580

1581

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”.

1582

1583

1584

1585

CONFIRM

1586

1587

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

1588

1589

Parameter

Type

target address

BTP address

inferior identifier	Identifier
qualifiers	List of qualifiers

1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617

target address the address to which the CONFIRM 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.

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).

Types of FAULT possible (sent to Superior address)

General

InvalidInferior – if inferior identifier is unknown

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

CONFIRMED

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 CONFIRM_ONE_PHASE if the Inferior decides to confirm its associated operations.

Parameter	Type
target address	BTP address
superior identifier	Identifier
inferior identifier	Identifier
confirm received	Boolean
qualifiers	List of qualifiers

1618
1619
1620
1621
1622
1623
1624

target address the address to which the CONFIRMED is sent. This will be the Superior address as on the CONTEXT message.

superior identifier the superior identifier as on the CONTEXT message.

inferior identifier the inferior identifier as on the earlier ENROL message.

1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641

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).

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.

1642
1643
1644
1645

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

1646
1647
1648
1649
1650
1651

The form CONFIRMED/auto refers to a CONFIRMED message with “confirm received” = “false”; CONFIRMED/response refers to a CONFIRMED message with “confirm received” = ”true”.

CANCEL

1652
1653
1654
1655

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

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

1656
1657
1658
1659
1660
1661
1662
1663

target address the address to which the CANCEL 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.

qualifiers standardised or other qualifiers.

1664 When received by an Inferior, the effects of any operations associated with the Inferior
1665 should be undone. If the Inferior had sent PREPARED, the Inferior is released from its
1666 promise to be able to confirm the operations.

1667
1668 Types of FAULT possible (sent to Superior address)

1669
1670 **General**

1671 **InvalidInferior** – if inferior identifier is unknown, or an inferior-handle
1672 on the inferiors-list is unknown

1673 **WrongState** – if a CONFIRM has been received by this Inferior.

1674
1675

1676 CANCELLED

1677
1678 Sent when the Inferior has applied (or is applying) cancellation of the operations associated
1679 with the Inferior. CANCELLED is sent from Inferior to Superior in the following cases:

- 1680
- 1681 1. before (and instead of) sending PREPARED, to indicate the Inferior is unable to
1682 apply the operations in full and is cancelling all of them;
 - 1683 2. in reply to CANCEL, regardless of whether PREPARED has been sent;
 - 1684 3. after sending PREPARED and then making and applying an autonomous
1685 decision to cancel.
 - 1686 4. in reply to CONFIRM_ONE_PHASE if the Inferior decides to cancel the
1687 associated operations
- 1688
1689
1690

1691
1692 As is specified in the state tables, cases 1, 2 and 3 are not always distinct in some
1693 circumstances of recovery and resending of messages.

1694

Parameter

target address	BTP address
superior identifier	Identifier
inferior identifier	Identifier
qualifiers	List of qualifiers

1695

1696 **target address** the address to which the CANCELLED is sent. This will be the
1697 Superior address as on the CONTEXT message.

1698

1699 **superior identifier** the superior identifier as on the CONTEXT message.

1700

1701 **inferior identifier** W the inferior identifier as on the earlier ENROL message.

1702

1703 **qualifiers** standardised or other qualifiers.

1704

1705 Types of FAULT possible (sent to address-as-inferior)

1706

1707 *General*

1708 *InvalidSuperior* – if Superior identifier is unknown

1709 *InvalidInferior* – if no ENROL has been received for this address-as-
1710 inferior and identifier, or if RESIGN has been received from this Inferior

1711 *WrongState* – if CONFIRM has been sent

1712

1713 Note – A CANCELLED message arriving before a CANCEL message is

1714 sent, or after a CONFIRM has been sent will occur when the Inferior has

1715 taken an autonomous decision and is not regarded as occurring in the wrong

1716 state. (The latter will cause a CONTRADICTION message to be sent.)

1717

1718

1719 CONFIRM_ONE_PHASE

1720

1721 Sent from a Superior to an enrolled Inferior, when there is only one such enrolled Inferior. In

1722 this case the two-phase exchange is not performed between the Superior and Inferior and the

1723 outcome decision for the operations associated with the Inferior is determined by the Inferior.

1724

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

1725

1726 **target address** the address to which the CONFIRM_ONE_PHASE message is
1727 sent This will be the address-as-inferior on the ENROL message.

1728

1729 **inferior identifier** The inferior identifier as on the earlier ENROL message for
1730 this Inferior.

1731

1732 **report hazard** Defines whether the superior wishes to be informed if a mixed
1733 condition occurs for the operations associated with the Inferior. If “report hazard”
1734 is “true”, the Inferior will reply with HAZARD if a mixed condition occurs, or if
1735 the Inferior cannot determine that a mixed condition has not occurred. If “report
1736 hazard” is false, the Inferior will report only its own decision, regardless of
1737 whether that decision was correctly and consistently applied. Default is false.

1738

1739 **qualifiers** standardised or other qualifiers.

1740
1741 CONFIRM_ONE_PHASE can be issued by a Superior to an Inferior from whom
1742 PREPARED has been received (subject to the requirement that there is only one enrolled
1743 Inferior).

1744
1745 Types of FAULT possible (sent to Superior address)

1746
1747 *General*
1748 *InvalidInferior* – if inferior identifier is unknown
1749 *WrongState* – if a PREPARE has already been sent to this Inferior

1750
1751 **HAZARD**

1752
1753 Sent when the Inferior has either discovered a “mixed” condition: that is unable to correctly
1754 and consistently cancel or confirm the operations in accord with the decision (either the
1755 received decision of the superior or its own autonomous decision), or when the Inferior is
1756 unable to determine that a “mixed” condition has not occurred.

1757
1758 HAZARD is also used to reply to a CONFIRM_ONE_PHASE if the Inferior determines there
1759 is a mixed condition within its associated operations or is unable to determine that there is not
1760 a mixed condition.

1761

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

1762
1763 **target address** the address to which the HAZARD is sent. This will be the
1764 superior address from the ENROL message.

1765
1766 **superior identifier** The superior identifier as on the ENROL message

1767
1768
1769 **inferior identifier** The inferior identifier as on the earlier ENROL message

1770
1771 **level** indicates, with value “mixed” that a mixed condition has definitely
1772 occurred; or, with value “possible” that it is unable to determine whether a mixed
1773 condition has occurred or not.

1774
1775 **qualifiers** standardised or other qualifiers.

1776
1777 Types of FAULT possible (sent to address-as-inferior)

1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794

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

1795
1796
1797
1798
1799
1800
1801
1802
1803

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)

1804
1805
1806
1807
1808
1809
1810

General

InvalidInferior – if inferior identifier is unknown

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

SUPERIOR_STATE

1811
1812
1813
1814
1815
1816
1817
1818

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).

1819
1820
1821
1822

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

1823
1824
1825
1826
1827
1828
1829
1830
1831
1832

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

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

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

status value	Meaning
<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

1833
1834
1835
1836
1837
1838
1839
1840

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

qualifiers standardised or other qualifiers.

1841 The Inferior, on receiving SUPERIOR_STATE with reply requested = true, should reply in a
1842 timely manner by (depending on its state) repeating the previous message it sent or by
1843 sending INFERIOR_STATE with the appropriate status value.

1844
1845 A status of unknown shall only be sent if it has been determined for certain that the Superior
1846 has no knowledge of the Inferior, or (equivalently) it can be determined that the relationship
1847 with the Inferior was cancelled. If there could be persistent information corresponding to the
1848 Superior, but it is not accessible from the entity receiving an INFERIOR_STATE/*y (or
1849 other) message targeted to the Superior or that entity cannot determine whether any such
1850 persistent information exists or not, the response shall be Inaccessible.

1851
1852 SUPERIOR_STATE/unknown is also used as a response to messages, other than
1853 INFERIOR_STATE/*y that are received when the Inferior is not known (and it is known
1854 there is no state information for it).

1855
1856 The form SUPERIOR_STATE/abcd refers to a SUPERIOR_STATE message status having a
1857 value equivalent to “abcd” (for active, prepared-received, unknown and inaccessible) and
1858 with “reply requested” = “false”. SUPERIOR_STATE/abcd/y refers to a similar message, but
1859 with “reply requested” = “true”. The form SUPERIOR_STATE/*y refers to a
1860 SUPERIOR_STATE message with “reply requested” = “true” and any value for status.

1861

1862

1863 INFERIOR_STATE

1864

1865 Sent by an Inferior as a query when in the active state to a Superior, when (due recovery from
1866 previous failure or other reason) there is uncertainty what state the Superior has reached.

1867

1868 Also sent by the Inferior to the Superior in response to a received SUPERIOR_STATE, in
1869 particular states.

1870

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

1871

1872 **target address** the address to which the INFERIOR_STATE is sent. This will
1873 be the target address as used the original ENROL message.

1874

1875 **superior identifier** The superior identifier as used on the ENROL message

1876

1877 **inferior identifier** The inferior identifier as on the ENROL message

1878
1879
1880
1881
1882

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

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
<i>unknown</i>	The Inferior is not known – it does not exist from the perspective of the Superior. The Inferior can be treated as cancelled

1883
1884
1885
1886
1887
1888

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”.

1889
1890

qualifiers standardised or other qualifiers.

1891
1892
1893
1894

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.

1895
1896
1897
1898
1899
1900
1901

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”.

1902
1903
1904
1905

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).

1906
1907
1908
1909
1910
1911
1912

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.

1913 The form INFERIOR_STATE/abcd refers to a INFERIOR_STATE message status having a
 1914 value equivalent to “abcd” (for active, unknown and inaccessible) and with “reply requested”
 1915 = “false”. INFERIOR_STATE/abcd/y refers to a similar message, but with “reply requested”
 1916 = “true”. The form INFERIOR_STATE/*y refers to a INFERIOR_STATE message with
 1917 “reply requested” = “true” and any value for status.
 1918
 1919

1920 REDIRECT

1921
 1922 Sent when the address previously given for a Superior or Inferior is no longer valid and the
 1923 relevant state information is now accessible with a different address (but the same superior or
 1924 inferior identifier).
 1925

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

1926
 1927 **target address** the address to which the REDIRECT is sent. This may be the
 1928 reply address from a received message or the address of the opposite side
 1929 (superior/inferior) as given in a CONTEXT or ENROL message
 1930

1931 **superior identifier** The superior identifier as on the CONTEXT message and
 1932 used on an ENROL message. (present only if the REDIRECT is sent from the
 1933 Inferior).
 1934

1935 **inferior identifier** The inferior identifier as on the ENROL message
 1936

1937 **old address** The previous address of the sender of REDIRECT. A match is
 1938 considered to apply if any of the old addresses match one that is already known.
 1939

1940 **new address** The (set of alternatives) new addresses to be used for messages
 1941 sent to this entity.
 1942

1943 **qualifiers** standardised or other qualifiers.
 1944

1945 If the actor whose address is changed is an Inferior, the new address value
 1946 replaces the address-as-inferior as present in the ENROL.
 1947

1948 If the actor whose address is changed is a Superior, the new address value
1949 replaces the Superior address as present in the CONTEXT message (or as present
1950 in any other mechanism used to establish the Superior:Inferior relationship).
1951
1952

1953 Messages used in control relationships

1954 BEGIN

1955
1956 A request to a Factory to create a new Business Transaction. This may either be a new top-
1957 level transaction, in which case the Composer or Coordinator will be the Decider, or the new
1958 Business Transaction may be immediately made the Inferior within an existing Business
1959 Transaction (thus creating a sub-Composer or sub-Coordinator).
1960
1961

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

1962
1963 **target address** the address of the entity to which the BEGIN is sent. How this
1964 address is acquired and the nature of the entity are outside the scope of this
1965 specification.
1966

1967 **reply address** the address to which the replying BEGUN and related
1968 CONTEXT message should be sent.
1969

1970 **transaction type** identifies whether a new Cohesion or new Atom is to be
1971 created; this value will be the “superior type” in the new CONTEXT
1972

1973 **qualifiers** standardised or other qualifiers. The standard qualifier “Transaction
1974 timelimit” may be present on BEGIN, to set the timelimit for the new business
1975 transaction and will be copied to the new CONTEXT. The standard qualifier
1976 “Inferior name” may be present if there is a CONTEXT related to the BEGIN.
1977

1978 A new top-level Business Transaction is created if there is no CONTEXT related to the
1979 BEGIN. A Business Transaction that is to be Inferior in an existing Business Transaction is
1980 created if the CONTEXT message for the existing Business Transaction is related to the
1981 BEGIN. In this case, the Factory is responsible for enrolling the new Composer or
1982 Coordinator as an Inferior of the Superior identified in that CONTEXT.
1983

1984
1985
1986

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.

1987
1988
1989

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

1990
1991

Types of FAULT possible (sent to Reply address)

1992
1993

General

1994
1995

BEGUN

1996
1997

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

1998
1999

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

2000
2001

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

2002
2003

2004
2005

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

2006
2007

2008
2009

2010
2011

address-as-inferior for a non-top-most transaction (a CONTEXT was related to the BEGIN), this is the address-as-inferior used in the enrolment with the Superior identified by the CONTEXT related to the BEGIN. The parameter is optional (implementor’s choice) if this is not a top-most transaction; it shall be absent if this is a top-most transaction this parameter.

2012
2013

2014
2015

2016
2017

transaction-identifier if this is a top-most transaction, this is an globally-unambiguous identifier for the new Decider (Composer or Coordinator). If this is not a top-most transaction, the transaction-identifier shall be the inferior-

2018

2019 identifier used in the enrolment with the Superior identified by the CONTEXT
2020 related to the BEGIN.
2021

2022 Note – The “transaction-identifier” may be identical to the “superior-
2023 identifier” in the CONTEXT that is related to the BEGUN

2024 **qualifiers** standardised or other qualifiers.

2025 At implementation option, the “address-as-decider” and/or “address-as-inferior” and the
2026 “address-as-superior” in the related CONTEXT may be the same or may be different. There
2027 is no general requirement that they even use the same bindings. Any may also be the same as
2028 the target address of the BEGIN message (the identifier on messages will ensure they are
2029 applied to the appropriate Composer or Coordinator).
2030
2031

2032 No FAULT messages are issued on receiving BEGUN.
2033
2034

2035 **PREPARE_INFERIORS**

2036 Sent from a Terminator to a Decider, but only if it is a Cohesion Composer, to tell it to
2037 prepare all or some of its inferiors, by sending PREPARE to any that have not already sent
2038 PREPARED, RESIGN or CANCELLED to the Decider (Composer) on its relationships as
2039 Superior. If the inferiors-list parameter is absent, the request applies to all the inferiors; if the
2040 parameter is present, it applies only to the identified inferiors of the Decider (Composer).
2041
2042

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

2043 **target address** the address to which the PREPARE_INFERIORS message is
2044 sent. This will be the decider-address from the BEGUN message.
2045
2046

2047 **reply address** the address of the Terminator sending the
2048 PREPARE_INFERIORS message.
2049

2050 **transaction identifier** identifies the Decider and will be the transaction-identifier
2051 from the BEGUN message.
2052

2053 **inferiors-list** defines which of the Inferiors of this Decider preparation is
2054 requested for, using the “inferior-identifiers” as on the ENROL received by the

2055 Decider (in its role as Superior). If this parameter is absent, the PREPARE
2056 applies to all Inferiors.

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

2059
2060
2061 For all Inferiors identified in the inferiors-list parameter (all Inferiors if the parameter is
2062 absent), from which none of PREPARED, CANCELLED or RESIGNED has been received,
2063 the Decider shall issue PREPARE. It will reply to the Terminator, using the reply address on
2064 the PREPARE_INFERIORS message, sending an INFERIOR_STATUSES message giving
2065 the status of the Inferiors identified on the inferiors-list parameter (all of them if the
2066 parameter was absent).

2067
2068 Types of FAULT possible (sent to Superior address)

2069
2070 *General*
2071 *InvalidDecider* – if Decider address is unknown
2072 *UnknownTransaction* – if the transaction-identifier is unknown
2073 *InvalidInferior* – if an inferior-handle on the inferiors-list is unknown
2074 *WrongState* – if a CONFIRM_TRANSACTION or
2075 CANCEL_TRANSACTION has already been received by this
2076 Composer.

2077
2078 The form PREPARE_INFERIORS/all refers to a PREPARE_INFERIORS message where
2079 the “inferiors-list” parameter is absent. The form PREPARE_INFERIORS/specific refers to a
2080 PREPARE_INFERIORS message where the “inferiors-list” parameter is present.

2081
2082
2083 **CONFIRM_TRANSACTION**

2084
2085 Sent from a Terminator to a Decider to request confirmation of the business transaction. If the
2086 business transaction is a Cohesion, the confirm-set is specified by the “inferiors-list”
2087 parameter.

Parameter	Type
target address	BTP address
reply address	BTP address
transaction identifier	Identifier
inferiors-list	List of Identifiers
report-hazard	Boolean
Qualifiers	List of qualifiers

2089

2090 **target address** the address to which the CONFIRM_TRANSACTION message
2091 is sent. This will be the address-as-decider on the BEGUN message.

2092
2093 **reply address** the address of the Terminator sending the
2094 CONFIRM_TRANSACTION message.

2095
2096 **transaction identifier** identifies the Decider. This will be the transaction-
2097 identifier from the BEGUN message.

2098
2099 **inferiors-list** defines which Inferiors enrolled with the Decider, if it is a
2100 Cohesion Composer, are to be confirmed, using the “inferior-identifiers” as on
2101 the ENROL received by the Decider (in its role as Superior). Shall be absent if
2102 the Decider is an Atom Coordinator.

2103
2104 **report hazard** Defines whether the Terminator wishes to be informed of hazard
2105 events and contradictory decisions within the business transaction. If “report
2106 hazard” is “true”, the receiver will wait until responses (CONFIRMED,
2107 CANCELLED or HAZARD) have been received from all of its inferiors,
2108 ensuring that any hazard events are reported. If “report hazard” is “false”, the
2109 Decider will reply with CONFIRM_COMPLETE or CANCEL_COMPLETE as
2110 soon as the decision for the transaction is known.

2111
2112 **qualifiers** standardised or other qualifiers.

2113
2114 If the “inferiors-list” parameter is present, the Inferiors identified shall be the “confirm-set” of
2115 the Cohesion. If the parameter is absent and the business transaction is a Cohesion, the
2116 “confirm-set” shall be all remaining Inferiors. If the business transaction is an Atom, the
2117 “confirm-set” is automatically all the Inferiors.

2118
2119 Any Inferiors from which RESIGN is received are not counted in the confirm-set.

2120
2121 If, for each of the Inferiors in the confirm-set, PREPARE has not been sent and PREPARED
2122 has not been received, PREPARE shall be issued to that Inferior.

2123

2124 NOTE -- If PREPARE has been sent but PREPARED not yet received from
2125 an Inferior in the confirm-set, it is an implementation option whether and
2126 when to re-send PREPARE. The Superior implementation may choose to re-
2127 send PREPARE if there are indications that the earlier PREPARE was not
2128 delivered.

2129
2130
2131 A confirm decision may be made only if PREPARED has been received from all Inferiors in
2132 the “confirm-set”. The making of the decision shall be persistent (and if it is not possible to
2133 persist the decision, it is not made). If there is only one remaining Inferior in the “confirm
2134 set” and PREPARE has not been sent to it, CONFIRM_ONE_PHASE may be sent to it.

2135 All remaining Inferiors that are not in the confirm set shall be cancelled.
 2136
 2137
 2138 If a confirm decision is made and “report-hazard” was “false”, a CONFIRM_COMPLETE
 2139 message shall be sent to the “reply-address”.
 2140
 2141 If a cancel decision is made and “report-hazard” was “false”, a CANCEL_COMPLETE
 2142 message shall be sent to the “reply-address”.
 2143
 2144 If “report-hazard” was “true” and any HAZARD or contradictory message was received (i.e.
 2145 CANCELLED from an Inferior in the confirm-set or CONFIRMED from an Inferior not in
 2146 the confirm-set), an INFERIOR_STATUSES reporting the status for all Inferiors shall be sent
 2147 to the “reply-address”.
 2148
 2149 Types of FAULT possible (sent to reply address)

2150
 2151 *General*

2152 *InvalidDecider* – if Decider address is unknown

2153 *UnknownTransaction* – if the transaction-identifier is unknown

2154 *InvalidInferior* – if an inferior handle in the inferiors-list is unknown

2155 *WrongState* – if a CANCEL_TRANSACTION has already been
 2156 received .
 2157

2158 The form CONFIRM_TRANSACTION/all refers to a CONFIRM_TRANSACTION message
 2159 where the “inferiors-list” parameter is absent. The form
 2160 CONFIRM_TRANSACTION/specific refers to a CONFIRM_TRANSACTION message
 2161 where the “inferiors-list” parameter is present.
 2162

2163 **TRANSACTION_CONFIRMED**

2164
 2165 A Decider sends TRANSACTION_CONFIRMED to a Terminator in reply to
 2166 CONFIRM_TRANSACTION if all of the confirm-set confirms (and, for a Cohesion, all other
 2167 Inferiors cancel) without reporting hazards, or if the Decider made a confirm decision and the
 2168 CONFIRM_TRANSACTION had a “report-hazards” value of “false”.
 2169

Parameter	Type
target address	BTP address
transaction-identifier	identifier
qualifiers	List of qualifiers

2170
 2171 **target address** the address to which the TRANSACTION_CONFIRMED is
 2172 sent., this will be the reply address from the CONFIRM_TRANSACTION
 2173 message.
 2174

2175 **transaction identifier** the transaction identifier as on the BEGUN message (i.e.
2176 the identifier of the Decider as a whole).

2177
2178 **qualifiers** standardised or other qualifiers.

2179
2180 Types of FAULT possible (sent to address-as-decider)

2181
2182 *General*
2183 *InvalidTerminator* – if Terminator address is unknown
2184 *UnknownTransaction* – if the transaction-identifier is unknown

2185
2186 **CANCEL_TRANSACTION**

2187
2188 Sent by a Terminator to a Decider at any time before CONFIRM_TRANSACTION has been
2189 sent.
2190

Parameter	Type
target address	BTP address
reply address	BTP address
transaction identifier	Identifier
report-hazard	Boolean
qualifiers	List of qualifiers

2191
2192 **target address** the address to which the CANCEL_TRANSACTION message is
2193 sent. This will be the decider-address from the BEGUN message.

2194
2195 **reply address** the address of the Terminator sending the
2196 CANCEL_TRANSACTION message.

2197
2198 **transaction identifier** identifies the Decider and will be the transaction-identifier
2199 from the BEGUN message.

2200
2201 **report hazard** Defines whether the Terminator wishes to be informed of hazard
2202 events and contradictory decisions within the business transaction. If “report
2203 hazard” is “true”, the receiver will wait until responses (CONFIRMED,
2204 CANCELLED or HAZARD) have been received from all of its inferiors,
2205 ensuring that any hazard events are reported. If “report hazard” is “false”, the
2206 Decider will reply with TRANSACTION_CANCELLED immediately.

2207
2208 **qualifiers** standardised or other qualifiers.

2209
2210 The business transaction is cancelled – this is propagated to any remaining Inferiors by
2211 issuing CANCEL to them. No more Inferiors will be permitted to enrol.
2212

2213 Types of FAULT possible (sent to Superior address)

2214

2215

General

2216

InvalidDecider – if Decider address is unknown

2217

UnknownTransaction – if the transaction-identifier is unknown

2218

WrongState – if a CONFIRM_TRANSACTION has been received by this Composer.

2219

2220

2221

2222

CANCEL_INFERIORS

2223

2224

Sent by a Terminator to a Decider, but only if is a Cohesion Composer, at any time before CONFIRM_TRANSACTION or CANCEL_TRANSACTION has been sent.

2225

2226

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

2227

2228

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

2229

2230

2231

reply address the address of the Terminator sending the CANCEL_TRANSACTION message.

2232

2233

2234

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

2235

2236

2237

inferiors-list defines which of the Inferiors of this Decider are to be cancelled, using the “inferior-identifiers” as on the ENROL received by the Decider (in its role as Superior).

2238

2239

2240

2241

qualifiers standardised or other qualifiers.

2242

2243

2244

Only the Inferiors identified in the inferiors-list are to be cancelled. Any other inferiors are unaffected by a CANCEL_INFERIORS. Further Inferiors may be enrolled.

2245

2246

2247

Note – A CANCEL_INFERIORS all of the currently enrolled Inferiors will leave the cohesion ‘empty’, but permitted to continue with new Inferiors, if any enrol.

2248

2249

2291 asking it about the status of that transaction tree nodes Inferiors, if there are any. In this latter
 2292 case, the receiver may reject the request with a FAULT(StatusRefused). If it is prepared to
 2293 reply, but has no Inferiors, it replies with an INFERIOR_STATUSES with an empty “status-
 2294 list” parameter.
 2295

Parameter	Type
target address	BTP address
reply address	BTP address
target-identifier	Identifier
inferiors-list	List of Identifiers
Qualifiers	List of qualifiers

2296
 2297 **target address** the address to which the REQUEST_STATUS message is sent.
 2298 When used to a Decider, this will be the address-as-decider from the BEGUN
 2299 message. Otherwise it may be an address-as-superior from a CONTEXT or
 2300 address-as-inferior from an ENROL message.
 2301

2302 **reply address** the address to which the replying INFERIOR_STATUSES is to
 2303 be sent
 2304

2305 **target-identifier** identifies the transaction (or transaction tree node) within the
 2306 scope of the target address. When the message is used to a Decider, this will be
 2307 the transaction-identifier from the BEGUN message. Otherwise it will be the
 2308 superior-identifier from a CONTEXT or an inferior-identifier from an ENROL
 2309 message.
 2310

2311 **inferiors-list** defines which inferiors enrolled with the target are to be included
 2312 in the INFERIOR_STATUSES, using the “inferior-identifiers” as on the ENROL
 2313 received by the Decider (in its role as Superior). If the list is absent, the status of
 2314 all enrolled Inferiors will be reported.
 2315

2316 **qualifiers** standardised or other qualifiers.
 2317

2318 Types of FAULT possible (sent to reply-address)
 2319

2320 *General*

2321 **StatusRefused** – if the receiver is not prepared to report its status to the
 2322 sender of this message. This FAULT type shall not be issued when a Decider
 2323 receives REQUEST_STATUSES from the Terminator.

2324 **UnknownTransaction** – if the transaction-identifier is unknown
 2325
 2326

2327 The form REQUEST_INFERIOR_STATUSES/all refers to a REQUEST_STATUS with the
 2328 inferiors-list absent. The form REQUEST_INFERIOR_STATUS/specific refers to a
 2329 REQUEST_INFERIOR_STATUS with the inferiors-list present.

2330

2331 **INFERIOR_STATUSES**

2332

2333 Sent by a Decider to report the status of all or some of its inferiors in response to a
 2334 REQUEST_INFERIOR_STATUSES, PREPARE_INFERIORS, CANCEL_INFERIORS,
 2335 CANCEL_TRANSACTION with “report-hazard” value of “true” and
 2336 CONFIRM_TRANSACTION with “report-hazard” value of “true”. It is also used by any
 2337 actor in response to a received REQUEST_INFERIOR_STATUSES to report the status of
 2338 inferiors, if there are any.

2339

Parameter	Type
target address	BTP address
responders-identifier	Identifier
status-list	Set of Status items - see below
general-qualifiers	List of qualifiers

2340

2341 **target address** the address to which the INFERIOR_STATUSES is sent. This
 2342 will be the reply address on the received message

2343

2344 **responders-identifier** the target-identifier used on the
 2345 REQUEST_INFERIOR_STATUSES.

2346

2347 **status-list** contains a number of Status-items, each reporting the status of one of
 2348 the inferiors of the Decider. The fields of a Status-item are

2349

Field	Type
Inferior-identifier	Inferior-identifier, 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).

2350

2351 The status value reports the current status of the particular inferior, as known to
 2352 the Decider (Composer or Coordinator). Values are:

2353

status value	Meaning
<i>active</i>	The Inferior is enrolled

status value	Meaning
<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
<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
<i>invalid</i>	No such inferior is enrolled (used only in reply to a REQUEST_INFERIOR_STATUSES/specific)

2354

2355

2356

2357

2358

2359

2360

2361

2362

2363

2364

2365

2366

2367

2368

2369

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

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

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

General

InvalidTerminator – if Terminator address is unknown

UnknownTransaction – if the transaction-identifier is unknown

2370
2371
2372
2373

2374 **Groups – combinations of related messages**

2375
2376
2377
2378
2379
2380
2381

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.

2382

2383 **CONTEXT & application message**

2384

2385
2386
2387
2388
2389
2390

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.

2391

2392
2393
2394
2395

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).

2396

2397
2398
2399
2400

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).

2401

2402
2403
2404
2405

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.

2406

2407
2408
2409

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.

2410

2411
2412
2413
2414

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.

2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461

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.

2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472

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.

2473
2474

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

2475
2476
2477
2478
2479

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).

2480
2481
2482

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

2483
2484
2485
2486

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.

2487
2488
2489
2490
2491

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).

2492
2493
2494
2495
2496

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.

2497
2498
2499

CONTEXT_REPLY & ENROL & application message (& PREPARED)

2500
2501
2502
2503
2504
2505
2506

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

2507 transmission) has been associated with the Inferior identified by the ENROL and will be
2508 subject to the outcome delivered to that Inferior.

2509
2510 **Target address:** the target address of the group is the target address of the
2511 CONTEXT_REPLY which shall also be the target address of the application message.
2512 The ENROL and PREPARED messages do not contain their target addresses.

2513
2514 The processing of ENROL and PREPARED messages is the same as for the previous
2515 groups.

2516
2517 This group can be used when participation in business transaction (normally a cohesion),
2518 is initiated by the service (Inferior) side, which fetches or acquires the CONTEXT, with
2519 some associated application semantic, performs some work for the transaction and sends
2520 an application message with a related ENROL. The CONTEXT_REPLY allows the
2521 addressing of the application (and the CONTEXT_REPLY) to be distinct from that of the
2522 Superior.

2523
2524 The actor receiving the group may associate the “inferior-identifier” received on the
2525 ENROL with the application message in a manner that is visible to the application
2526 receiving the message (e.g. for subsequent use in Terminator:Decider exchanges).

2527

2528 **BEGUN & CONTEXT**

2529

2530 **Meaning:** the CONTEXT is that for the new business transaction, containing the
2531 Superior address.

2532

2533 **Target address:** the target address is that of the BEGUN message – this will be the reply
2534 address of the earlier BEGIN message.

2535

2536 **BEGIN & CONTEXT**

2537

2538 **Meaning:** the new business transaction is to be an Inferior (sub-coordinator or sub-
2539 composer) of the Superior identified by the CONTEXT. The Factory (receiver of the
2540 BEGIN) will perform the enrolment.

2541

2542 **Target address:** the target address is that of the BEGIN – this will be the address of the
2543 Factory.

2544

2545 **Standard qualifiers**

2546

2547 The following qualifiers are expected to be of general use to many applications and
2548 environments. The URI “urn:oasis:names:tc:BTP:qualifiers” is used in the
2549 Qualifier group value for the qualifiers defined here.

2550

2551

2552 **Transaction timelimit**

2553

2554 The transaction **timelimit** allows the Superior (or an application element initiating the
2555 business transaction) to indicate the expected length of the active phase, and thus give an
2556 indication to the Inferior of when it would be appropriate to initiate cancellation if the active
2557 phase appears to continue too long. The time limit ends (the clock stops) when the Inferior
2558 decides to be prepared and issues **PREPARED** to the Superior.
2559

2560 It should be noted that the expiry of the time limit does not change the permissible actions of
2561 the Inferior. At any time prior to deciding to be prepared (for an Inferior), the Inferior is
2562 **permitted** to initiate cancellation for internal reasons. The **timelimit** gives an indication to the
2563 entity of when it will be useful to exercise this right.
2564

2565 The qualifier is propagated on a **CONTEXT** message.
2566

2567 The “Qualifier name” shall be “**transaction-timelimit**” .
2568

2569 The “Content” shall contain the following field:
2570

Content field	Type
Timelimit	Integer

2571
2572 **Timelimit** indicates the maximum (further) duration, expressed as whole seconds from the
2573 time of transmission of the containing **CONTEXT**, of the active phase of the business
2574 transaction.
2575

2576 **Inferior timeout**

2577

2578 This qualifier allows an Inferior to limit the duration of its “promise”, when sending
2579 **PREPARED**, that it will maintain the ability to confirm or cancel the effects of all associated
2580 operations. Without this qualifier, an Inferior is expected to retain the ability to confirm or
2581 cancel indefinitely. If the timeout does expire, the Inferior is released from its promise and
2582 can apply the decision indicated in the qualifier.
2583

2584 It should be noted that BTP recognises the possibility that an Inferior may be forced to apply
2585 a confirm or cancel decision before the **CONFIRM** or **CANCEL** is received and before this
2586 timeout expires (or if this qualifier is not used). Such a decision is termed a heuristic decision,
2587 and (as with other transaction mechanisms), is considered to be an exceptional event. As with
2588 heuristic decisions, the taking of an autonomous decision by a Inferior **subsequent** to the
2589 expiry of this timeout, is liable to cause contradictory decisions across the business
2590 transaction. BTP ensures that at least the occurrence of such a contradiction will be
2591 (eventually) reported to the Superior of the business transaction. BTP treats “true” heuristic
2592 decisions and autonomous decisions after timeout the same way – in fact, the expiry in this
2593 timeout does not cause a qualitative (state table) change in what can happen, but rather a step
2594 change in the probability that it will.
2595

2596 The expiry of the timeout does not strictly require that the Inferior immediately invokes the
2597 intended decision, only that is at liberty to do so. An implementation may choose to only

2598 apply the decision if there is contention for the underlying resource, for example.
2599 Nevertheless, Superiors are recommended to avoid relying on this and ensure decisions for
2600 the business transaction are made before these timeouts expire (and allow a margin of error
2601 for network latency etc.).

2602
2603 The qualifier may be present on a PREPARED message. If the PREPARED message has the
2604 “default is cancel” parameter “true”, then the “IntendedDecision” field of this qualifier shall
2605 have the value “cancel”.

2606
2607 The “Qualifier name” shall be “inferior-timeout” .

2608
2609 The “Content” shall contain the following fields:
2610

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

2611
2612 **Timeout** indicates how long, expressed as whole seconds from the time of transmission of the
2613 carrying message, the Inferior intends to maintain its ability to either confirm or cancel the
2614 effects of the associated operations, as ordered by the receiving Superior.

2615
2616 **IntendedDecision** indicates which outcome will be applied, if the timeout completes and an
2617 autonomous decision is made.

2618
2619 **Minimum inferior timeout**

2620
2621 This qualifier allows a Superior to constrain the Inferior timeout qualifier received from the
2622 Inferior. If a Superior knows that the decision for the business transaction will not be
2623 determined for some period, it can require that Inferiors do not send PREPARED messages
2624 with Inferior timeouts that would expire before then. An Inferior that is unable or unwilling to
2625 send a PREPARED message with a longer (or no) timeout **should** cancel, and reply with
2626 CANCELLED.

2627
2628 The qualifier may be present on a CONTEXT, ENROLLED or PREPARE message. If
2629 present on more than one, and with different values of the MinimumTimeout field, the value
2630 on ENROLLED shall prevail over that on CONTEXT and the value on PREPARE shall
2631 prevail over either of the others.

2632
2633 The “Qualifier name” shall be “minimum-inferior-timeout” .

2634
2635 The “Content” shall contain the following field:
2636

Content field	Type
MinimumTimeout	Integer

2637

2638 **Minimum Timeout** is the minimum value of timeout, expressed as whole seconds, that will be
2639 acceptable in the Inferior timeout qualifier on an answering PREPARED message.

2640
2641 **Inferior name**

2642
2643 This qualifier allows an Enroller to supply a name for the Inferior that will be visible on
2644 INFERIOR_STATUSES and thus allow the Terminator to determine which Inferior (of the
2645 Composer or Coordinator) is related to which application work. This is in addition to the
2646 “inferior-identifier” field. The name can be human-readable and can also be used in fault
2647 tracing, debugging and auditing.

2648
2649 The name is never used by the BTP actors themselves to identify each other or to direct
2650 messages. (The BTP actors use the addresses and the identifiers in the message parameters
2651 for those purposes.)

2652
2653 This specification makes no requirement that the names are unambiguous within any scope
2654 (unlike the globally unambiguous “inferior-identifier” on ENROLLED and BEGUN). Other
2655 specifications, including those defining use of BTP with a particular application may place
2656 requirements on the use and form of the names. (This may include reference to information
2657 passed in application messages or in other, non-standardised, qualifiers.)

2658
2659 The qualifier may be present on BEGIN, ENROL and in the “qualifiers” field of a Status-item
2660 in INFERIOR_STATUSES. It is present on BEGIN only if there is a related CONTEXT; if
2661 present, the same qualifier value **should** be included in the consequent ENROL. If
2662 INFERIOR_STATUSES includes a Status-item for an Inferior whose ENROL had an
2663 inferior-name qualifier, the same qualifier value **should** be included in the Status-item.

2664
2665 The “Qualifier -name” shall be “inferior-name”

2666
2667 The “Content” shall contain the following fields:

Content field	Type
inferior-name	String

2669
2670 **Inferior name** the name assigned to the enrolling Inferior.
2671

2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717

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

2718 business transaction and on features of the implementation (e.g. making of a persistent record
2719 of the decision means that the information will survive at least some failures that otherwise
2720 lose state information, but the level of survival depends on the purpose of the
2721 implementation). [Table 2](#) and [Table 3](#) define the decision events.

2722
2723 In some cases, an implementation may not need to make an active change to have a persistent
2724 record of a decision, provided that the implementation will restore itself to the appropriate
2725 state on recovery. For example, an (inferior) implementation that “decided to be prepared”,
2726 and recorded a timeout (to cancel) in the persistent information for that decision (signalled via
2727 the appropriate qualifier on PREPARED), could treat the presence of an expired record as a
2728 record of “decide to cancel autonomously”, provided it always updated such a record as part
2729 of the “apply ordered confirmation” decision event.

2730
2731 The Superior event “decide to prepare” is considered semi-persistent. Since the sending of
2732 PREPARE indicates that the application exchange (to associate operations with the Inferior)
2733 is complete, it is not meaningful for the Superior:Inferior relationship to revert to an earlier
2734 state corresponding to an incomplete application exchange. However, implementations are
2735 not required to make the sending of PREPARE persistent in terms of recovery – a Superior
2736 that experiences failure after sending PREPARE may, on recovery, have no information
2737 about the transaction, in which case it is considered to be in the completed state (Z), which
2738 will imply the cancellation of the Inferior and its associated operations.

2739
2740 Where a Superior is itself an Inferior (to another Superior entity), in a hierarchic tree, its
2741 “decide to confirm” and “decide to cancel” decisions will in fact be the receipt of a
2742 CONFIRM or CANCEL instruction from its own Superior, without necessary change of local
2743 persistent information (which would combine both superior and inferior information, pointing
2744 both up and down the tree).

2745
2746

2747 **Disruptions – failure events**

2748
2749 Failure events are modelled as “disruption”. A failure and the subsequent recovery will (or
2750 may) cause a change of state. The disruption events in the state tables model different extents
2751 of loss of state information. An implementation is not required to exhibit all the possible
2752 disruption events, but it is not allowed to exhibit state transitions that do not correspond to a
2753 possible disruption.

2754
2755 In addition to the disruption events in the tables, there is an implicit “disruption 0” event,
2756 which involves possible interruption of service and loss of messages in transit, but no change
2757 of state (either because no state information was lost, or because recovery from persistent
2758 information restores the implementation to the same state). The “disruption 0” event would
2759 typically be an appropriate abstraction for a communication failure.

2760
2761

2762 **Invalid cells and assumptions of the communication mechanism**

2763
2764 The empty cells in state table represent events that cannot happen. For events corresponding
to sending a message or any of the decision events, this prohibition is absolute – e.g. a

2765 conformant implementation in the Superior active state “B1” will not send CONFIRM. For
2766 events corresponding to receiving a message, the interpretation depends on the properties of
2767 the underlying communications mechanism.

2768
2769 For all communication mechanisms, it is assumed that
2770 a) the two directions of the Superior:Inferior communication are not synchronised –
2771 that is messages travelling in opposite directions can cross each other to any
2772 degree; any number of messages may be in transit in either direction; and
2773 b) messages may be lost arbitrarily
2774

2775 If the communication mechanisms guarantee ordered delivery (i.e. that messages, if delivered
2776 at all, are delivered to the receiver in the order they were sent) , then receipt of a message in a
2777 state where the corresponding cell is empty indicates that the far-side has sent a message out
2778 of order – a FAULT message with the Fault Type “WrongState” can be returned.
2779

2780 If the communication mechanisms cannot guarantee ordered delivery, then messages received
2781 where the corresponding cell is empty should be ignored. Assuming the far-side is
2782 conformant, these messages can assumed to be “stale” and have been overtaken by messages
2783 sent later but already delivered. (If the far-side is non-conformant, there is a problem
2784 anyway).
2785

2786 **Meaning of state table events**

2787
2788 The tables in this section define the events (rows) in the state tables. [Table 1](#) defines
2789 the events corresponding to sending or receiving BTP messages and the disruption events.
2790 [Table 2](#) describes the decision events for an Inferior, [Table 3](#) those for a
2791 Superior.
2792

2793 The decision events for a Superior, defined in [Table 3](#) cannot be specified without
2794 reference to other Inferiors to which it is Superior and to its relation with the application or
2795 other entity that (acting ultimately on behalf of the application) drives it.
2796

2797 The term “remaining Inferiors” refers to any actors to which this endpoint is Superior and
2798 which are to be treated as an atomic decision unit with (and thus including) the Inferior on
2799 this relationship. If the CONTEXT for this Superior:Inferior relationship had a “superior
2800 type” of “atom”, this will be all Inferiors established with same Superior address and Superior
2801 identifier except those from which RESIGN has been received. If the CONTEXT had
2802 “superior type” of “cohesion”, the “remaining Inferiors” excludes any that it has been
2803 determined will be cancelled, as well as any that have resigned – in other words it includes
2804 only those for which a confirm decision is still possible or has been made. The determination
2805 of exactly which Inferiors are “remaining Inferiors” in a cohesion is determined, in some
2806 way, by the application. The term “Other remaining Inferiors” excludes this Inferior on this
2807 relationship. A Superior with a single Inferior will have no “other remaining Inferiors”.
2808

2809 In order to ensure that the confirmation decision is delivered to all remaining Inferiors,
2810 despite failures, the Superior must persistently record which these Inferiors are (i.e. their
2811 addresses and identifiers). It must also either record that the decision is confirm, or ensure

2812 that the confirm decision (if there is one) is persistently recorded somewhere else, and that it
 2813 will be told about it. This latter would apply if the Superior were also BTP Inferior to another
 2814 entity which persisted a confirm decision (or recursively deferred it still higher). However,
 2815 since there is no requirement that the Superior be also a BTP Inferior to any other entity, the
 2816 behaviour of asking another entity to make (and persist) the confirm decision is termed
 2817 "offering confirmation" - the Superior offers the possible confirmation of itself, and its
 2818 remaining Inferiors to some other entity. If that entity (or something higher up) then does
 2819 make and persist a confirm decision, the Superior is "instructed to confirm" (which is
 2820 equivalent BTP CONFIRM).

2821
 2822 The application, or an entity acting indirectly on behalf of the application, may request a
 2823 Superior to prepare an Inferior (or all Inferiors). This typically implies that there will be no
 2824 more operations associated with the Inferior. Following a request to prepare all remaining
 2825 Inferiors, the Superior may offer confirmation to the entity that requested the prepare. (If the
 2826 Superior is also a BTP Inferior, its superior can be considered an entity acting on behalf of the
 2827 application.)

2828
 2829 The application, or an entity acting indirectly on behalf of the application, may also request
 2830 confirmation. This means the Superior is to attempt to make and persist a confirm decision
 2831 itself, rather than offer confirmation.

2832
 2833

2834

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

2835

2836

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)

2837

2838

Table 3: Decision events for a Superior

Event name	Meaning
decide to confirm one-phase	<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 • If an atom, 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)

2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867

Persistent information

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.

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.

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.

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	one-phase confirm 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	CONFIRM_ONE_PHASE received after prepared state
s2	CONFIRM_ONE_PHASE received
s3	CONFIRM_ONE_PHASE received, confirming
s4	CONFIRM_ONE_PHASE received, cancelling
s5	CONFIRM_ONE_PHASE received, hazard detected
s6	CONFIRM_ONE_PHASE 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

2871

2872

2873

2874

2875

2876

The changes to the state tables are marked by colour, rather than change marks
Green = issue 81, for resending ENROL/rsp-req
Blue = issue 81, for resending ENROL/no-rsp-req
Orange = issue 104

Table 6: Superior state table – normal forward progression

	I 1	A1	B1	B2	C1	D1	E1	E2	F1	F2
recei ve ENROL/rsp-req	A1	A1	B2	B2		D1				
recei ve ENROL/no-rsp-req	B1		B1	B1		D1				
recei ve RESI GN/rsp-req	Y1		C1	C1	C1	C1				
recei ve RESI GN/no-rsp-req	Z		Z	Z	Z	Z				
recei ve PREPARED	Y1		E1	E1		E1	E1		F1	
recei ve PREPARED/cancel	Y1		E2	E2		E2		E2	F1	
recei ve CONFIR MED/auto	Q1		H1	H1		H1	H1		F1	
recei ve CONFIR MED/response									F2	F2
recei ve CANCELLED	Y1		Z	Z		Z	J1	J1	K1	
recei ve HAZARD	P1	P1	P1	P1		P1	P1	P1	P3	
recei ve INF_STATE/acti ve/y	Y1	A1	B1	B2		D1				
recei ve INF_STATE/acti ve			B1	B2		D1				
recei ve INF_STATE/unknown			Z	Z	Z	Z				
send ENROLLED		B1		B1						
send RESI GNED					Z					
send PREPARE						D1	E1	E2		
send CONFIR M_ONE_PHASE										
send CONFIR M									F1	
send CANCEL										
send CONTRADI CTI ON										
send SUP_STATE/acti ve/y			B1							
send SUP_STATE/acti ve			B1							
send SUP_STATE/prepared-rcvd/y							E1	E2		
send SUP_STATE/prepared-rcvd							E1	E2		
send SUP_STATE/unknown										
deci de to confi rm one-phase			S1	S1			S1	S1		
deci de to prepare			D1	D1						
deci de to confi rm			G1	G1			F1	F1		
deci de to cancel						G1	G1	Z		
remove persi stent i nformati on										Z
record contradi cti on										
di srupti on I	Z	Z	Z	Z	B1	Z	Z	Z		F1
di srupti on II					Z		D1	D1		
di srupti on III							B1	B1		
di srupti on IV										

2877

2878

Table 7: Superior state table – cancellation and contradiction

	G1	G2	G3	G4	H1	J1	K1	L1
recei ve ENROL/rsp-req	G1	G2						
recei ve ENROL/no-rsp-req	G1	G2						
recei ve RESI GN/rsp-req	G3	Z	G3					
recei ve RESI GN/no-rsp-req	Z	Z	Z					
recei ve PREPARED	G1	G2						
recei ve PREPARED/cancel	G1	G2						
recei ve CONFIR MED/auto	L1	L1			H1			L1
recei ve CONFIR MED/response								
recei ve CANCELLED	G4	Z		G4		J1	K1	
recei ve HAZARD	P4	P4						
recei ve INF_STATE/acti ve/y	G1	G2						
recei ve INF_STATE/acti ve	G1	G2						
recei ve INF_STATE/unknown	Z	Z	Z	Z				
send ENROLLED								
send RESI GNED								
send PREPARE								
send CONFIR M_ONE_PHASE								
send CONFIR M								
send CANCEL	G2	G2	Z	Z				
send CONTRADI CTI ON								
send SUP_STATE/acti ve/y								
send SUP_STATE/acti ve								
send SUP_STATE/prepared-rcvd/y								
send SUP_STATE/prepared-rcvd								
send SUP_STATE/unknown								
deci de to confi rm one-phase								
deci de to prepare					F1	K1		
deci de to confi rm					L1	G4		
deci de to cancel								
remove persi stent i nformati on							R1	R1
record contradi cti on								
di srupti on I	Z	Z	Z	Z	Z	Z	F1	Z
di srupti on II			G2	G2	E1	E1		G2
di srupti on III					D1	D1		
di srupti on IV					B1	B1		

Table 8: Superior state table – hazard and request confirm

	P1	P2	P3	P4	Q1	R1	R2	S1
recei ve ENROL/rsp-req								S1
recei ve ENROL/no-rsp-req								S1
recei ve RESI GN/rsp-req								Z
recei ve RESI GN/no-rsp-req								Z
recei ve PREPARED								S1
recei ve PREPARED/cancel								S1
recei ve CONFIR MED/auto					Q1	R1	R1	S1
recei ve CONFIR MED/response					Z	R2		Z
recei ve CANCELLED						R1	R1	Z
recei ve HAZARD	P1	P2	P3	P4		R1	R1	Z
recei ve INF_STATE/acti ve/y								S1
recei ve INF_STATE/acti ve								S1
recei ve INF_STATE/unknown	P1	P2		P4		R2	R2	Z
send ENROLLED								
send RESI GNED								
send PREPARE								
send CONFIR M_ONE_PHASE								S1
send CONFIR M								
send CANCEL								
send CONTRADI CTI ON						R2		
send SUP_STATE/acti ve/y								
send SUP_STATE/acti ve								
send SUP_STATE/prepared-rcvd/y								
send SUP_STATE/prepared-rcvd								
send SUP_STATE/unknown								
deci de to confi rm one-phase								
deci de to prepare								
deci de to confi rm								
deci de to cancel								
remove persi stent i nformati on							Z	
record contradi cti on	R1	R1	R1	R1	R1			
di srupti on I	Z	Z	Z	Z	Z		R1	Z
di srupti on II	D1		F1	G2				
di srupti on III	B1							
di srupti on IV								

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

	Y1	Z
recei ve ENROL/rsp-req	Y1	Y1
recei ve ENROL/no-rsp-req	Y1	Y1
recei ve RESI GN/rsp-req	Y1	Y1
recei ve RESI GN/no-rsp-req	Z	Z
recei ve PREPARED	Y1	Y1
recei ve PREPARED/cancel	Y1	Y1
recei ve CONFIR MED/auto	Q1	Q1
recei ve CONFIR MED/response	Z	Z
recei ve CANCELLED	Y1	Y1
recei ve HAZARD	P2	P2
recei ve INF_STATE/acti ve/y	Y1	Y1
recei ve INF_STATE/acti ve	Y1	Z
recei ve INF_STATE/unknown	Z	Z
send ENROLLED		
send RESI GNED		
send PREPARE		
send CONFIR M_ONE_PHASE		
send CONFIR M		
send CANCEL		
send CONTRADI CTI ON		
send SUP_STATE/acti ve/y		
send SUP_STATE/acti ve		
send SUP_STATE/prepared-rcvd/y		
send SUP_STATE/prepared-rcvd		
send SUP_STATE/unknown	Z	
deci de to confi rm one-phase		
deci de to prepare		
deci de to confi rm		
deci de to cancel		
remove persi stent i nformati on		
record contradi cti on		
di srupti on I	Z	
di srupti on II		
di srupti on III		
di srupti on IV		

2883

2884

Table 10: Inferior state table – normal forward progression

	i 1	a1	b1	c1	d1	e1	e2	f1	f2
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	a1 b1	a1	b1	c1 z		e1	e2		
send INF_STATE/active/y send INF_STATE/active send INF_STATE/unknown		a1	b1 b1		d1 d1				
receive ENROLLED receive RESIGNED receive PREPARE receive CONFIRM_ONE_PHASE receive CONFIRM receive CANCEL receive CONTRADICTION		b1	b1	c1 z		e1	e2		
receive SUP_STATE/active/y receive SUP_STATE/active receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd receive SUP_STATE/unknown		b1 b1	b1 b1	c1 c1		e1 e1	e2 e2		
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			c1 e1 e2		c1 e1 e2	h1 j1	z1	m1 m1	m1 m1
disruption I disruption II disruption III		z	z	z	z b1			e1	e2

2885

2886

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	g1	g2	h1 h1 h2 h2 l1 l2		j1 j1 k1 j2 j2 k2		k1 k2 k2		l1 l2 l2	
receive SUP_STATE/active/y receive SUP_STATE/active receive SUP_STATE/prepared-rcvd/y receive SUP_STATE/prepared-rcvd receive SUP_STATE/unknown	x1	x2	h1 h1 h1 h1 l1		j1 j1 j1 j1 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	n1 p2	n1 p2	m1		z		z		z	
disruption I disruption II disruption III	e1	e2	h1		j1		j1 k1 j1		h1 l1 h1	

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 RESI GN/rsp-req send RESI GN/no-rsp-req send PREPARED send PREPARED/cancel send CONFIR MED/auto send CONFIR MED/response send CANCELLED send HAZARD	z	z	p1	p2	q1
send INF_STATE/active/y send INF_STATE/active send INF_STATE/unknown					
recei ve ENROLLED recei ve RESI GNED recei ve PREPARE recei ve CONFIR M_ONE_PHASE recei ve CONFIR M recei ve CANCEL recei ve CONTRADI CTI ON	m1	n1	p1 s5 z	p2 s5 z	q1 s6 q1 q1 z
recei ve SUP_STATE/active/y recei ve SUP_STATE/active recei ve SUP_STATE/prepared-rcvd/y recei ve SUP_STATE/prepared-rcvd recei ve SUP_STATE/unknown		z	p1 p1 p1	p2 p2 p2	q1 q1 q1 q1
deci de to resi gn deci de to be prepared deci de to be prepared/cancel deci de to confi rm autonomously deci de to cancel autonomously appl y ordered confi rmati on remove persi stent i nformati on detect probl em detect and record probl em					q1 q1
di srupti on I di srupti on II di srupti on III	z	z d1 b1	z		

Table 13: Inferior state table – request confirm states

	s1	s2	s3	s4	s5	s6
send ENROL/rsp-req send ENROL/no-rsp-req send RESI GN/rsp-req send RESI GN/no-rsp-req send PREPARED send PREPARED/cancel send CONFIR MED/auto send CONFIR MED/response send CANCELLED send HAZARD			z	z	z	z
send INF_STATE/active/y send INF_STATE/active send INF_STATE/unknown						
recei ve ENROLLED recei ve RESI GNED recei ve PREPARE recei ve CONFIR M_ONE_PHASE recei ve CONFIR M recei ve CANCEL recei ve CONTRADI CTI ON	s1	s2	s3	s4	s5	s6
recei ve SUP_STATE/active/y recei ve SUP_STATE/active recei ve SUP_STATE/prepared-rcvd/y recei ve SUP_STATE/prepared-rcvd recei ve SUP_STATE/unknown	x1	z	z	z	z	z
deci de to resi gn deci de to be prepared deci de to be prepared/cancel deci de to confi rm autonomously deci de to cancel autonomously apply ordered confi rmati on remove persi stent i nformati on detect probl em detect and record probl em			s3 s4			s6
di srupti on I di srupti on II di srupti on III	e1	z		z	z	

2892

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 RESI GN/rsp-req send RESI GN/no-rsp-req send PREPARED send PREPARED/cancel send CONFIR MED/auto send CONFIR MED/response send CANCELLED send HAZARD						z1
send INF_STATE/active/y send INF_STATE/active send INF_STATE/unknown			z			
recei ve ENROLLED recei ve RESI GNED recei ve PREPARE recei ve CONFIR M_ONE_PHASE recei ve CONFIR M recei ve CANCEL recei ve CONTRADI CTI ON			y1 y1 y1 y1 y1 y1 z	y2 y2 y2 y2 z z	z z y1 y1 m1 y1 z	z1 z1 y1 y1 y2 y1 z
recei ve SUP_STATE/active/y recei ve SUP_STATE/active recei ve SUP_STATE/prepared-rcvd/y recei ve SUP_STATE/prepared-rcvd recei ve SUP_STATE/unknown			y1 y1 x1	y2 y2 y2 y2 x2	y1 z y2 z	y2 z1 y2 y2 z
deci de to resi gn deci de to be prepared deci de to be prepared/cancel deci de to confi rm autonomously deci de to cancel autonomously apply ordered confi rmati on remove persi stent i nformati on detect probl em detect and record probl em						
di srupti on I di srupti on II di srupti on III	e1	e2				

2893

2894

2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939

Failure Recovery

Types of failure

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

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

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

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

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

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

2940 After recovery from node failure, the implementation behaves much as if a communication
2941 failure had occurred.

2942

2943 Persistent information

2944

2945 BTP requires that some decision events are persisted – that information recording an
2946 Inferior’s decision to be prepared, a Superior’s decision to confirm and an Inferior’s
2947 autonomous decision survive failure. Making the first two decisions persistent ensures that a
2948 consistent decision can be reached for the business transaction and that it is delivered to all
2949 involved nodes. Requiring an Inferior’s autonomous decision to be persistent allows BTP to
2950 ensure that, if this decision is contradictory (i.e. opposite to the decision at the Superior), the
2951 contradiction will be reported to the Superior, despite failures.

2952

2953 BTP also permits, but does not require, recovery of the Superior:Inferior relationship in the
2954 active state (unlike many transaction protocols, where a communication or endpoint failure in
2955 active state would invariably cause rollback of the transaction). Recovery in the active state
2956 may require that the application exchange is resynchronised as well – BTP does not directly
2957 support this, but does allow continuation of the business transaction as such. In the state
2958 tables, from some states, there are several levels of disruption, distinguished by which state
2959 the implementation transits to – this represents the survival of different extents of state
2960 information over failure and recovery. The different levels of disruption describe legitimate
2961 states for the endpoint to be in after it has recovered – **they do not require that all
2962 implementations are able to exhibit the appropriate partial loss of state information.**

2963

2964 The absence of a destination state for the disruption events means that such a transition is not
2965 legitimate – thus, for example, an Inferior that has decided to be prepared will always recover
2966 to the same state, by virtue of the information persisted in the “decide to be prepared” event.

2966

2967 Apart from the (optional) recovery in active state, BTP follows the well-known presume-
2968 abort model – it is only required that information be persisted when decisions are made (and
2969 not, e.g. on enrolment). This means that on recovery, one side may have persistent
2970 information but the other does not. This occurs when an Inferior has decided to be prepared
2971 but the Superior never confirmed (so the decision is “presumed” to be cancel), or because the
2972 Superior did confirm, and the Inferior applied the confirm, removed its persistent information
2973 but the acknowledgement (CONFIRMED) was never received by the Superior (or, at least, it
2974 still had the persistent information when the failure occurred).

2975

2976 Information to be persisted for an Inferior’s “decision to be prepared” must be sufficient to
2977 re-establish communication with the Superior, to apply a confirm decision and to apply a
2978 cancel decision. It will thus need to include

2979

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

2980

Superior address (as on CONTEXT)

2981

Superior identifier (as on CONTEXT)

2982

default-is-cancel value (as on PREPARED)

2983

2984 The information needed to apply confirm/cancel decisions will depend on the application and
2985 the associated operations. It may also normally be necessary to persist any qualifiers that

2986 were sent with the PREPARED message or application messages sent with the PREPARED,
2987 since the PREPARED message will be repeated if a failure occurs.

2988

2989 A Superior must record corresponding information to allow it to re-establish communication
2990 with the Inferior:

2991 Inferior address (as on ENROL)

2992 Inferior identifier (as on ENROL)

2993

2994 A Superior that is the Decider for the business transaction need only persist this information
2995 if it makes a decision to confirm (and this Inferior is in the confirm set, for a Cohesion). A
2996 Superior that is also an Inferior to some other entity (i.e. it is an intermediate in a tree, as
2997 atom in a cohesion, sub-coordinator or sub-composer) must persist this information as
2998 Superior (to this Inferior) as part of the persistent information of its decision to be prepared
2999 (as an Inferior). For such an entity, the “decision to confirm” as Superior is made when (and
3000 if) CONFIRM is received from its Superior or it makes an autonomous decision to confirm. If
3001 CONFIRM is received, the persistent information may be changed to show the confirm
3002 decision, but alternatively, the receipt of the CONFIRM can be treated as the decision itself.
3003 If the persistent information is left unchanged and there is a node failure, on recovery the
3004 entity (as an Inferior) will be in a prepared state, and will rediscover the confirm decision
3005 (using the recovery exchanges to its Superior) before propagating it to its Inferior(s).

3006

3007 After failure, an implementation may not be able to restore an endpoint to the appropriate
3008 state immediately – in particular, the necessary persistent information may be inaccessible,
3009 although the implementation can respond to received BTP messages. In such a case, a
3010 Superior may reply to any BTP message except INFERIOR_STATE/* (i.e. with a “reply-
3011 requested” value “false”) with SUPERIOR_STATE/inaccessible and an Inferior to any BTP
3012 message except SUPERIOR_STATE/* with “INFERIOR_STATE/inaccessible. Receipt of
3013 the *_STATE/inaccessible messages has no effect on the endpoint state.

3014

3015 Redirection

3016

3017 As described above, BTP uses the presume-abort model for recovery. A corollary of this is
3018 that there are cases where one side will attempt to re-establish communication when there is
3019 no persistent information for the relationship at the far-end. In such cases, it is important the
3020 side that is attempting recovery can distinguish between unsuccessful attempts to connect to
3021 the holder of the persistent information and when the information no longer exists. If the peer
3022 information does not exist, this side can draw conclusions and complete appropriately; if they
3023 merely fail to get through they are stuck in attempting recovery.

3024

3025 Two mechanisms are provided to make it possible that even when one side of a
3026 Superior:Inferior relationship has completed, that a message can eventually get through to
3027 something that can definitively report the status, distinguishing this case from a temporary
3028 inability to access the state of a continuing transaction element. The mechanisms are:

- 3029 o Address fields which provide a “callback address” can be a set of addresses,
3030 which are alternatives one of which is chosen as the target address for the
3031 future message. If the sender of that message finds the address does not work,
3032 it can try a different alternative.

3033 o The REDIRECT message can be used to inform the peer that an address
3034 previously given is no longer valid and to supply a replacement address (or
3035 set of addresses). REDIRECT can be issued either as a response to receipt of
3036 a message or spontaneously.

3037
3038 The two mechanisms can be used in combination, with one or more of the original set of
3039 addresses just being a redirector, which does not itself ever have direct access to the state
3040 information for the transaction, but will respond to any message with an appropriate
3041 REDIRECT.

3042
3043 An alternative implementation approach is to have a single addressable entity that uses the
3044 same address for all transactions, distinguishing them by identifier, and which always
3045 recovers to use the same address. Such an implementation would not need to supply
3046 “backup” addresses (and would only use REDIRECT if it was being permanently migrated).

3047
3048
3049

Terminator:Decider failures

3050 BTP does not provide facilities or impose requirements on the recovery of
3051 Terminator:Decider relationships, other than allowing messages to be repeated. A Terminator
3052 may survive failures (by retaining knowledge of the Decider’s address and identifier), but this
3053 is an implementation option. Although a Decider (if it decides to confirm) will persist
3054 information about the confirm decision, it is not required, after failure, to remain accessible
3055 using the inferior address it offered to the Terminator. Any such recovery is an
3056 implementation option.

3057
3058 A Decider’s address (as returned on BEGUN) may be a set of addresses, allowing a failed
3059 Decider to be recovered at a different address.

3060
3061 A Decider has no way of initiating a call to a Terminator to ensure that it is still active, and
3062 thus no way of detecting that a Terminator has failed. To avoid a Decider waiting for ever for
3063 a CONFIRM_TRANSACTION that will never arrive, the standard qualifier “Transaction
3064 timelimit” can be used (by the Initiator) to inform the Decider when it can assume the
3065 Terminator will not issue CONFIRM_TRANSACTION and so it (the Decider) should initiate
3066 cancellation.

3067

3068 XML representation of Message Set

3069
3070 This section describes the syntax for BTP messages in XML. These XML messages represent
3071 a midpoint between the abstract messages and what actually gets sent on the wire.

3072
3073 All BTP related URIs have been created using Oasis URI conventions as specified in [RFC](#)
3074 [3121](#)

3075
3076 The XML Namespace for the BTP messages is urn:oasis:names:tc:BTP:xml

3077
3078 In addition to an XML schema, this specification uses an informal syntax to describe the
3079 structure of the BTP messages. The syntax appears as an XML instance, but the values

3080 contain data types instead of values. The following symbols are appended to some of the
3081 XML constructs: ? (zero or one), * (zero or more), + (one or more.) The absence of one of
3082 these symbols corresponds to "one and only one."
3083

3084 Addresses

3085
3086 As described in the "Abstract Message and Associated Contracts – Addresses" section, a BTP
3087 address comprises three parts, and for a target address only the "additional information" field
3088 is inside the BTP messages. For all BTP messages whose abstract form includes a target
3089 address parameter, the corresponding XML representation includes a "target-additional-
3090 information" element. This element may be omitted if it would be empty.
3091

3092 For other addresses, all three fields are represent, as in:

```
3093 <ctp:some-address>  
3094   <ctp:binding-name>...carrier binding URI...</ctp:binding-name>  
3095   <ctp:binding-address>...carrier specific  
3096   address...</ctp:binding-address>  
3097   <ctp:additional-information>...optional additional addressing  
3098   information...</ctp:additional-information> ?  
3099 </ctp:some-address>
```

3100
3101
3102 A "published" address can be a set of <some-address>, which are alternatives which can be
3103 chosen by the peer (sender.) Multiple addresses are used in two cases: different bindings to
3104 same endpoint, or backup endpoints. In the former, the receiver of the message has the choice
3105 of which address to use (depending on which binding is preferable.) In the case where
3106 multiple addresses are used for redundancy, a priority attribute can be specified to help the
3107 receiver choose among the addresses- the address with the highest priority should be used,
3108 other things being equal. The priority is used as a hint and does not enforce any behaviour in
3109 the receiver of the message. Default priority is a value of 1.
3110

3111 Qualifiers

3112 The "Qualifier name" is used as the element name, within the namespace of the "Qualifier
3113 group".
3114
3115

3116 Examples:

```
3117 <ctpq:inferior-timeout  
3118   xmlns:ctpq="urn:oasis:names:tc:BTP:qualifiers"  
3119   xmlns:ctp="urn:oasis:names:tc:BTP:xml "  
3120   ctp:must-be-understood="false"  
3121   ctp:to-be-propagated="false">1800</ctpq:inferior-timeout>  
3122  
3123 <auth:username  
3124   xmlns:auth="http://www.example.com/ns/auth"  
3125   xmlns:ctp="urn:oasis:names:tc:BTP:xml "  
3126   ctp:must-be-understood="true"  
3127   ctp:to-be-propagated="true">jtauber</auth:username>
```

3129 Attributes must-be-understood **has default value “true”** and to-be-propagated has default
3130 value “false”.

3131

3132 Identifiers

3133

3134 Identifiers shall be URIs "

3135

3136 Note – Identifiers need to be globally unambiguous. Apart from their
3137 generation, the only operation the BTP implementations have to perform on
3138 identifiers is to match them.

3139

3140 Message References

3141 Each BTP message has an optional id attribute to give it a unique identifier. An application
3142 can make use of those identifiers, but no processing is enforced.

3143

3144 Messages

3145

3146 CONTEXT

3147

```
3148 <btpt:context id?>  
3149 <btpt:superior-address> +  
3150 ...address...  
3151 </btpt:superior-address>  
3152 <btpt:superior-identifier>...URI...</btpt:superior-identifier>  
3153 <btpt:reply-address> ?  
3154 ...address...  
3155 </btpt:reply-address>  
3156 <btpt:superior-type>cohesion|atom</btpt:superior-type>  
3157 <btpt:qualifiers> ?  
3158 ...qualifiers...  
3159 </btpt:qualifiers>  
3160 </btpt:context>
```

3161

3162 CONTEXT_REPLY

3163

```
3164 <btpt:context-reply id?>  
3165 <btpt:target-additional-information> ?  
3166 ...additional address information...  
3167 </btpt:target-additional-information>  
  
3168 <btpt:superior-identifier>...URI...</btpt:superior-identifier>  
3169 <btpt:completion-  
3170 status>completed|related|repudiated</btpt:completion-status>  
3171 <btpt:qualifiers> ?  
3172 ...qualifiers...  
3173 </btpt:qualifiers>  
3174 </btpt:context-reply>
```

3175

3176

3177
3178
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3220
3221
3222
3223
3224
3225
3226
3227

REQUEST_STATUS

```
<btpr:request-status id?>  
  <btpr:target-additional-information> ?  
    ...additional address information...  
</btpr:target-additional-information>  
  <btpr:reply-address> ?  
    ...address...  
</btpr:reply-address>  
  <btpr:target-identifier>...URI...</btpr:target-identifier>  
  <btpr:qualifiers> ?  
    ...qualifiers...  
</btpr:qualifiers>  
</btpr:request-status>
```

STATUS

```
<btpr:status id?>  
  <btpr:target-additional-information> ?  
    ...additional address information...  
</btpr:target-additional-information>  
  <btpr:responders-identifier>...URI...</btpr:responders-identifier>  
  
  <btpr:status-value>created|enrolling|active|resigning|  
    resigned|preparing|prepared|  
    confirming|confirmed|cancelling|cancelled|  
    cancel-contradiction|confirm-contradiction|  
    hazard|contradicted|unknown|inaccessible</btpr:status-  
value>  
  <btpr:qualifiers> ?  
    ...qualifiers...  
</btpr:qualifiers>  
</btpr:status>
```

FAULT

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

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

- 3228
- 3229 o communication-failure
- 3230 o duplicate-inferior
- 3231 o general
- 3232 o invalid-decider
- 3233 o invalid-inferior
- 3234 o invalid-superior
- 3235 o status-refused
- 3236 o invalid-terminator
- 3237 o unknown-parameter
- 3238 o unknown-transaction
- 3239 o unsupported-qualifier
- 3240 o wrong-state

3241

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

3245

Fault data can take on various forms:

3246

Free text:

3247

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

3251

Identifier:

3252

```
<btp: fault-data>...URI...</btp: fault-data>
```

3255

3256

Inferior Identity:

3257

```
<btp: fault-data>
  <btp: inferior-address> +
    ...address...
  </btp: inferior-address>
  <btp: inferior-identifier>...URI...</btp: inferior-identifier>
</btp: fault-data>
```

3265

ENROL

3266

```
<btp: enrol       id?>
  <btp: target-additional-information> ?
    ...additional address information...
  </btp: target-additional-information>
  <btp: superior-identifier>...URI...</btp: superior-identifier>
  <btp: reply-requested>true|false</btp: reply-requested>
  <btp: reply-address> ?
    ...address...
```

3275

```
3276 </btp:reply-address>
3277 <btp:inferior-address> +
3278 ...address...
3279 </btp:inferior-address>
3280 <btp:inferior-identifier>...URI...</btp:inferior-identifier>
3281 <btp:qualifiers> ?
3282 ...qualifiers...
3283 </btp:qualifiers>
3284 </btp:enrol>
```

3285
3286

ENROLLED

```
3287
3288
3289 <btp:enrolled id?>
3290 <btp:target-additional-information> ?
3291 ...additional address information...
3292 </btp:target-additional-information>
3293 <btp:inferior-identifier>...URI...</btp:inferior-identifier>
3294 <btp:qualifiers> ?
3295 ...qualifiers...
3296 </btp:qualifiers>
3297 </btp:enrolled>
```

3298
3299

RESIGN

```
3300
3301
3302 <btp:resign id?>
3303 <btp:target-additional-information> ?
3304 ...additional address information...
3305 </btp:target-additional-information>
3306 <btp:superior-identifier>...URI...</btp:superior-identifier>
3307 <btp:inferior-identifier>...URI...</btp:inferior-identifier>
3308 <btp:response-requested>true|false</btp:response-requested>
3309 <btp:qualifiers> ?
3310 ...qualifiers...
3311 </btp:qualifiers>
3312 </btp:resign>
```

3313
3314

RESIGNED

```
3315
3316
3317 <btp:resigned id?>
3318 <btp:target-additional-information> ?
3319 ...additional address information...
3320 </btp:target-additional-information>
3321 <btp:inferior-identifier>...URI...</btp:inferior-identifier>
3322 <btp:qualifiers> ?
3323 ...qualifiers...
3324 </btp:qualifiers>
3325 </btp:resigned>
```

3326

3327
3328
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3370
3371
3372
3373
3374
3375
3376
3377

PREPARE

```
<btpp:prepare id?>  
  <btpp:target-additional-information> ?  
    ...additional address information...  
  </btpp:target-additional-information>  
  <btpp:inferior-identifier>...URI...</btpp:inferior-identifier>  
  <btpp:qualifiers> ?  
    ...qualifiers...  
  </btpp:qualifiers>  
</btpp:prepare>
```

PREPARED

```
<btpp:prepared id?>  
  <btpp:target-additional-information> ?  
    ...additional address information...  
  </btpp:target-additional-information>  
  <btpp:superior-identifier>...URI...</btpp:superior-identifier>  
  <btpp:inferior-identifier>...URI...</btpp:inferior-identifier>  
  <btpp:default-is-cancel>true|false</btpp:default-is-cancel>  
  <btpp:qualifiers> ?  
    ...qualifiers...  
  </btpp:qualifiers>  
</btpp:prepared>
```

CONFIRM

```
<btpp:confirm id?>  
  <btpp:target-additional-information> ?  
    ...additional address information...  
  </btpp:target-additional-information>  
  <btpp:inferior-identifier>...URI...</btpp:inferior-identifier>  
  <btpp:qualifiers> ?  
    ...qualifiers...  
  </btpp:qualifiers>  
</btpp:confirm>
```

CONFIRMED

```
<btpp:confirmed id?>  
  <btpp:target-additional-information> ?  
    ...additional address information...  
  </btpp:target-additional-information>  
  <btpp:superior-identifier>...URI...</btpp:superior-identifier>  
  <btpp:inferior-identifier>...URI...</btpp:inferior-identifier>  
  <btpp:confirmed-received>true|false</btpp:confirmed-received>
```

```
3378 <btpr:qualifiers> ?
3379   ...qualifiers...
3380 </btpr:qualifiers>
3381 </btpr:confirmed>
```

CANCEL

```
3385
3386 <btpr:cancel id?>
3387   <btpr:target-additional-information> ?
3388     ...additional address information...
3389   </btpr:target-additional-information>
3390   <btpr:inferior-identifier>...URI...</btpr:inferior-identifier>
3391   <btpr:reply-address> ?
3392     ...address...
3393   </btpr:reply-address>
3394   <btpr:qualifiers> ?
3395     ...qualifiers...
3396   </btpr:qualifiers>
3397 </btpr:cancel>
```

CANCELLED

```
3400
3401
3402 <btpr:cancelled id?>
3403   <btpr:target-additional-information> ?
3404     ...additional address information...
3405   </btpr:target-additional-information>
3406   <btpr:superior-identifier>...URI...</btpr:superior-identifier>
3407
3408   <btpr:inferior-identifier>...URI...</btpr:inferior-identifier> ?
3409   <btpr:qualifiers> ?
3410     ...qualifiers...
3411   </btpr:qualifiers>
3412 </btpr:cancelled>
```

CONFIRM_ONE_PHASE

```
3413
3414
3415
3416
3417 <btpr:confirm-one-phase id?>
3418   <btpr:target-additional-information> ?
3419     ...additional address information...
3420   </btpr:target-additional-information>
3421   <btpr:inferior-identifier>...URI...</btpr:inferior-identifier>
3422   <btpr:report-hazard>true|false</btpr:report-hazard>
3423   <btpr:qualifiers> ?
3424     ...qualifiers...
3425   </btpr:qualifiers>
3426 </btpr:confirm-one-phase>
3427
```

3428
3429
3430
3431
3432
3433
3434
3435
3436
3437
3438
3439
3440
3441
3442
3443

HAZARD

```
<btpt:hazard id?>  
  <btpt:target-additional-information> ?  
    ...additional address information...  
  </btpt:target-additional-information>  
  <btpt:superior-identifier>...URI...</btpt:superior-identifier>  
  
  <btpt:inferior-identifier>...URI...</btpt:inferior-identifier>  
  <btpt:level>mixed|possible</btpt:level>  
  <btpt:qualifiers> ?  
    ...qualifiers...  
  </btpt:qualifiers>  
</btpt:hazard>
```

3444
3445
3446
3447
3448
3449
3450
3451
3452
3453
3454
3455

CONTRADICTION

```
<btpt:contradiction id?>  
  <btpt:target-additional-information> ?  
    ...additional address information...  
  </btpt:target-additional-information>  
  <btpt:inferior-identifier>...URI...</btpt:inferior-identifier>  
  <btpt:qualifiers> ?  
    ...qualifiers...  
  </btpt:qualifiers>  
</btpt:contradiction>
```

3456
3457

SUPERIOR_STATE

3458
3459
3460
3461
3462
3463
3464
3465
3466
3467
3468
3469
3470
3471
3472

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

3473
3474
3475
3476
3477
3478

INFERIOR_STATE

```
<btpt:inferior-state id?>  
  <btpt:target-additional-information> ?  
    ...additional address information...  
  </btpt:target-additional-information>
```

```
3479 <btpr:superior-identifier>...URI...</btpr:superior-identifier>
3480
3481 <btpr:inferior-identifier>...URI...</btpr:inferior-identifier>
3482 <btpr:status>active|inaccessible|unknown</btpr:status>
3483 <btpr:reply-requested>true|false</btpr:reply-requested>
3484 <btpr:qualifiers> ?
3485     ...qualifiers...
3486 </btpr:qualifiers>
3487 </btpr:inferior-state>
```

REDIRECT

```
3491 <btpr:redirect id?>
3492   <btpr:target-additional-information> ?
3493     ...additional address information...
3494 </btpr:target-additional-information>
3495   <btpr:superior-identifier>...URI...</btpr:superior-identifier> ?
3496   <btpr:inferior-identifier>...URI...</btpr:inferior-identifier>
3497   <btpr:old-address> +
3498     ...address...
3499 </btpr:old-address>
3500   <btpr:new-address> +
3501     ...address...
3502 </btpr:new-address>
3503   <btpr:qualifiers> ?
3504     ...qualifiers...
3505 </btpr:qualifiers>
3506 </btpr:redirect>
```

BEGIN

```
3510 <btpr:begin id?>
3511   <btpr:target-additional-information> ?
3512     ...additional address information...
3513 </btpr:target-additional-information>
3514   <btpr:reply-address> ?
3515     ...address...
3516 </btpr:reply-address>
3517   <btpr:transaction-type>cohesion|atom</btpr:transaction-type>
3518   <btpr:qualifiers> ?
3519     ...qualifiers...
3520 </btpr:qualifiers>
3521 </btpr:begin>
```

BEGUN

```
3522 <btpr:begin id?>
3523   <btpr:target-additional-information> ?
3524     ...additional address information...
```

```

3530     </btp:target-additional-information>
3531     <btp:decider-address> *
3532         ...address...
3533     </btp:decider-address>
3534     <btp:inferior-address> *
3535         ...address...
3536     </btp:inferior-address>
3537     <btp:transaction-identifier>...URI...</btp:transaction-
3538 identifier>
3539     <btp:qualifiers> ?
3540         ...qualifiers...
3541     </btp:qualifiers>
3542 </btp:begin>

```

3543
3544

PREPARE_INFERIORS

```

3546     <btp:prepare-inferiors id?>
3547     <btp:target-additional-information> ?
3548         ...additional address information...
3549     </btp:target-additional-information>
3550     <btp:reply-address> ?
3551         ...address...
3552     </btp:reply-address>
3553     <btp:transaction-identifier>...URI...</btp:transaction-
3554 identifier>
3555     <btp:inferiors-list> ?
3556         <btp:inferior-handle>...URI...</btp:inferior-handle> +
3557     </btp:inferiors-list>
3558     <btp:qualifiers> ?
3559         ...qualifiers...
3560     </btp:qualifiers>
3561 </btp:prepare-inferiors>

```

3563
3564

CONFIRM_TRANSACTION

```

3566     <btp:confirm-transaction id?>
3567     <btp:target-additional-information> ?
3568         ...additional address information...
3569     </btp:target-additional-information>
3570     <btp:reply-address> ?
3571         ...address...
3572     </btp:reply-address>
3573     <btp:transaction-identifier>...URI...</btp:transaction-
3574 identifier>
3575     <btp:inferiors-list> ?
3576         <btp:inferior-handle>...URI...</btp:inferior-handle> +
3577     </btp:inferiors-list>
3578     <btp:report-hazard>true|false</btp:report-hazard>
3579     <btp:qualifiers> ?
3580         ...qualifiers...
3581

```

```
3582     </btp:qualifiers>
3583 </btp:confirm_transaction>
```

TRANSACTION_CONFIRMED

```
3587
3588 <btp:transaction-confirmed id?>
3589   <btp:target-additional-information> ?
3590     ...additional address information...
3591   </btp:target-additional-information>
3592
3593   <btp:transaction-identifier>...URI...</btp:transaction-
3594   identifier>
3595   <btp:qualifiers> ?
3596     ...qualifiers...
3597   </btp:qualifiers>
3598 </btp:transaction-confirmed>
```

CANCEL_TRANSACTION

```
3601
3602
3603 <btp:cancel-transaction id?>
3604   <btp:target-additional-information> ?
3605     ...additional address information...
3606   </btp:target-additional-information>
3607   <btp:reply-address> ?
3608     ...address...
3609   </btp:reply-address>
3610   <btp:transaction-identifier>...URI...</btp:transaction-
3611   identifier>
3612   <btp:report-hazard>true|false</btp:report-hazard>
3613   <btp:qualifiers> ?
3614     ...qualifiers...
3615   </btp:qualifiers>
3616 </btp:cancel-transaction>
```

CANCEL_INFERIORS

```
3617
3618
3619
3620 <btp:cancel-inferiors id?>
3621   <btp:target-additional-information> ?
3622     ...additional address information...
3623   </btp:target-additional-information>
3624   <btp:reply-address> ?
3625     ...address...
3626   </btp:reply-address>
3627   <btp:transaction-identifier>...URI...</btp:transaction-
3628   identifier> ?
3629   <btp:inferiors-list>
3630     <btp:inferior-handle>...URI...</btp:inferior-handle> +
3631   </btp:inferiors-list>
3632   <btp:qualifiers> ?
```

```
3633     ...qualifiers...
3634     </btp:qualifiers>
3635 </btp:cancel-inferiors>
```

3636
3637

TRANSACTION_CANCELLED

```
3639
3640 <btp:transaction-cancelled id?>
3641   <btp:target-additional-information> ?
3642     ...additional address information...
3643   </btp:target-additional-information>
3644
3645   <btp:transaction-identifier>...URI...</btp:transaction-
3646   identifier>
3647   <btp:qualifiers> ?
3648     ...qualifiers...
3649   </btp:qualifiers>
3650 </btp:transaction-cancelled>
```

3651
3652

REQUEST_INFERIOR_STATUSES

```
3654
3655 <btp:request-inferior-statuses id?>
3656   <btp:target-additional-information> ?
3657     ...additional address information...
3658   </btp:target-additional-information>
3659   <btp:reply-address> ?
3660     ...address...
3661   </btp:reply-address>
3662   <btp:target-identifier>...URI...</btp:target-identifier>
3663   <btp:inferiors-list> ?
3664     <btp:inferior-handle>...URI...</btp:inferior-handle> +
3665   </btp:inferiors-list>
3666   <btp:qualifiers> ?
3667     ...qualifiers...
3668   </btp:qualifiers>
3669 </btp:request-inferior-statuses>
```

3670
3671

INFERIOR_STATUSES

```
3672
3673
3674 <btp:inferior-statuses id?>
3675   <btp:target-additional-information> ?
3676     ...additional address information...
3677   </btp:target-additional-information>
3678
3679   <btp:responders-identifier>...URI...</btp:responders-identifier>
3680   <btp:status-list>
3681     <btp:status-item> +
3682       <btp:inferior-handle>...URI...</btp:inferior-handle>
3683       <btp:status>active|resigned|preparing|prepared|
```

```

3684         autonomously-confirmed|autonomously-cancelled|
3685         confirming|confirmed|cancelling|cancelled|
3686         cancel-contradiction|confirm-contradiction|
3687         hazard|invalid</btp:status>
3688     <btp:qualifiers> ?
3689         ...qualifiers...
3690 </btp:qualifiers>
3691 </btp:status-item>
3692 </btp:status-list>
3693 <btp:qualifiers> ?
3694     ...qualifiers...
3695 </btp:qualifiers>
3696 </btp:inferior-statuses>
3697

```

3698 **Standard qualifiers**

3699 The informal syntax for these messages assumes the namespace prefix “btpq” is associated
3700 with the URI “urn:oasis:names:tc:BTP:qualifiers”.

3702 **Transaction timelimit**

```

3704 <btpq:transaction-timelimit>
3705     <btpq:timelimit>
3706         ...time in seconds...
3707     </btpq:timelimit>
3708 </btpq:transaction-timelimit>
3709

```

3710 **Inferior timeout**

```

3711 <btpq:inferior-timeout>
3712     <btpq:timeout>
3713         ...time in seconds...
3714     </btpq:timeout>
3715     <btpq:intended-decision>confirm|cancel</btpq:intended-decision>
3716 </btpq:inferior-timeout>
3717

```

3718 **Minimum inferior timeout**

```

3719 <btpq:minimum-inferior-timeout>
3720     <btpq:minimum-timeout>
3721         ...time in seconds...
3722     </btpq:minimum-timeout>
3723 </btpq:minimum-inferior-timeout>
3724

```

3725 **Inferior name**

```

3726 <btpq:inferior-name>
3727     <btpq:inferior-name>
3728         ...string...
3729     </btpq:inferior-name>
3730 </btpq:inferior-name>
3731

```

3732 **Compounding of Messages**

3733

3734 Relating BTP to one another, in a “group” is represented by containing them within the
3735 btp:related-group element, with the related messages as child elements. The processing for
3736 the group is defined in the section “Groups – combinations of related messages”. For example

3737
3738
3739
3740
3741
3742
3743
3744
3745

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

3746 If the rules for the group state that the target address of the abstract message is omitted, the
3747 corresponding target-address-information element shall be absent in the message in the
3748 related-group. The carrier protocol binding specifies how a relation between application and
3749 BTP messages is represented.

3750

3751 Bundling (semantically insignificant combination) of BTP messages and related groups is
3752 indicated with the "btp:messages" element, with the bundled messages and related groups as
3753 child elements. For example (confirming one and cancelling another inferiors of a cohesion):

3754
3755
3756
3757
3758
3759
3760
3761

```
<btp:messages>  
  <btp:confirm>...</btp:confirm>  
  <btp:cancel>...</btp:cancel>  
</btp:messages>
```

3761

XML Schemas

3762

3763

3764

XML schema for BTP messages

3765

3766

```
<?xml version="1.0"?>
```

3767

```
<schema
```

3768

```
  xmlns="http://www.w3.org/2001/XMLSchema"
```

3769

```
  targetNamespace="urn:oasis:names:tc:BTP:xml"
```

3770

```
  xmlns:btp="urn:oasis:names:tc:BTP:xml"
```

3771

```
  elementFormDefault="qualified">
```

3772

3773

3774

```
  <!-- Qualifiers -->
```

3775

3776

```
  <complexType name="qualifier-type">
```

3777

```
    <simpleContent>
```

3778

```
      <extension base="string">
```

3779

```
        <attribute name="must-be-understood" type="boolean"/>
```

3780

```
        <attribute name="to-be-propagated" type="boolean"/>
```

3781

```
      </extension>
```

3782

```
    </simpleContent>
```

3783

```
  </complexType>
```

3784

3785

```
  <element name="qualifier" type="btp:qualifier-type" abstract="true"/>
```

3786

3787

```
  <element name="qualifiers">
```

3788

```
    <complexType>
```

3789

```
      <sequence>
```

3790

```
        <element ref="btp:qualifier" maxOccurs="unbounded"/>
```

3791

```
      </sequence>
```

3792

```
    </complexType>
```

3793

```
  </element>
```

3794

3795

```
  <!-- example qualifier:
```

3796

```
    <element name="some-qualifer" type="btp:qualifier-type"
```

3797

```
  substitutionGroup="btp:qualifier"/>
```

3798

```
  -->
```

3799

3800

3801

```
  <!-- Message set data types -->
```

3802

3803

```
  <simpleType name="identifier">
```

3804

```
    <restriction base="anyURI" />
```

3805

```
  </simpleType>
```

3806

3807

```
  <simpleType name="additional-information">
```

3808

```
    <restriction base="string" />
```

3809

```
  </simpleType>
```

3810

3811

```
  <complexType name="address">
```

3812

```
    <sequence>
```

```

3813         <element name="binding-name" type="anyURI"/>
3814         <element name="binding-address" type="string"/>
3815         <element name="additional-information" type="btp:additional-
3816 information" minOccurs="0" />
3817     </sequence>
3818 </complexType>
3819
3820     <simpleType name="superior-type">
3821         <restriction base="string">
3822             <enumeration value="cohesion"/>
3823             <enumeration value="atom"/>
3824         </restriction>
3825     </simpleType>
3826
3827     <simpleType name="transaction-type">
3828         <restriction base="string">
3829             <enumeration value="cohesion"/>
3830             <enumeration value="atom"/>
3831         </restriction>
3832     </simpleType>
3833
3834
3835     <!-- Compounding -->
3836
3837     <element name="messages">
3838         <complexType>
3839             <sequence>
3840                 <element ref="btp:message" minOccurs="0"
3841 maxOccurs="unbounded"/>
3842             </sequence>
3843         </complexType>
3844     </element>
3845
3846     <element name="related-group" substitutionGroup="btp:message">
3847         <complexType>
3848             <sequence>
3849                 <element ref="btp:message" minOccurs="0"
3850 maxOccurs="unbounded"/>
3851             </sequence>
3852         </complexType>
3853     </element>
3854
3855
3856     <!-- Message set -->
3857
3858     <element name="message" abstract="true" />
3859
3860     <element name="context" substitutionGroup="btp:message">
3861         <complexType>
3862             <sequence>
3863                 <element name="superior-address" type="btp:address"
3864 maxOccurs="unbounded"/>
3865                 <element name="superior-identifier" type="btp:identifier"/>

```

```

3866         <element name="reply-address" type="btp:address"
3867 minOccurs="0"/>
3868         <element name="superior-type" type="btp:superior-type"/>
3869         <element ref="btp:qualifiers" minOccurs="0"/>
3870     </sequence>
3871     <attribute name="id" type="ID" use="optional"/>
3872 </complexType>
3873 </element>
3874
3875 <element name="context-reply" substitutionGroup="btp:message">
3876 <complexType>
3877 <sequence>
3878     <element name="target-additional-information"
3879 type="btp:additional-information" minOccurs="0"/>
3880     <element name="superior-identifier" type="btp:identifier"/>
3881     <element name="completion-status">
3882 <simpleType>
3883 <restriction base="string">
3884     <enumeration value="completed"/>
3885     <enumeration value="related"/>
3886     <enumeration value="repudiated"/>
3887 </restriction>
3888 </simpleType>
3889 </element>
3890     <element ref="btp:qualifiers" minOccurs="0"/>
3891 </sequence>
3892 <attribute name="id" type="ID"/>
3893 </complexType>
3894 </element>
3895
3896 <element name="request-status" substitutionGroup="btp:message">
3897 <complexType>
3898 <sequence>
3899     <element name="target-additional-information"
3900 type="btp:additional-information" minOccurs="0"/>
3901     <element name="reply-address" type="btp:address"
3902 minOccurs="0"/>
3903     <element name="target-identifier" type="btp:identifier"/>
3904     <element ref="btp:qualifiers" minOccurs="0"/>
3905 </sequence>
3906 <attribute name="id" type="ID"/>
3907 </complexType>
3908 </element>
3909
3910 <element name="status" substitutionGroup="btp:message">
3911 <complexType>
3912 <sequence>
3913     <element name="target-additional-information"
3914 type="btp:additional-information" minOccurs="0"/>
3915     <element name="responders-identifier"
3916 type="btp:identifier"/>
3917     <element name="status-value">
3918 <simpleType>

```

```

3919         <restriction base="string">
3920             <enumeration value="created"/>
3921             <enumeration value="enrolling"/>
3922             <enumeration value="active"/>
3923             <enumeration value="resigning"/>
3924             <enumeration value="resigned"/>
3925             <enumeration value="preparing"/>
3926             <enumeration value="prepared"/>
3927             <enumeration value="confirming"/>
3928             <enumeration value="confirmed"/>
3929             <enumeration value="cancelling"/>
3930             <enumeration value="cancelled"/>
3931             <enumeration value="cancel-contradiction"/>
3932             <enumeration value="confirm-contradiction"/>
3933             <enumeration value="hazard"/>
3934             <enumeration value="contradicted"/>
3935             <enumeration value="unknown"/>
3936             <enumeration value="inaccessible"/>
3937         </restriction>
3938     </simpleType>
3939 </element>
3940     <element ref="btp:qualifiers" minOccurs="0"/>
3941 </sequence>
3942 <attribute name="id" type="ID"/>
3943 </complexType>
3944 </element>
3945
3946 <element name="fault" substitutionGroup="btp:message">
3947     <complexType>
3948         <sequence>
3949             <element name="target-additional-information"
3950 type="btp:additional-information" minOccurs="0"/>
3951             <element name="superior-identifier" type="btp:identifier"
3952 minOccurs="0"/>
3953             <element name="inferior-identifier" type="btp:identifier"
3954 minOccurs="0"/>
3955             <element name="fault-type">
3956                 <simpleType>
3957                     <restriction base="string">
3958                         <enumeration value="communication-failure"/>
3959                         <enumeration value="duplicate-inferior"/>
3960                         <enumeration value="general"/>
3961                         <enumeration value="invalid-decider"/>
3962                         <enumeration value="invalid-inferior"/>
3963                         <enumeration value="invalid-superior"/>
3964                         <enumeration value="status-refused"/>
3965                         <enumeration value="invalid-terminator"/>
3966                         <enumeration value="unknown-parameter"/>
3967                         <enumeration value="unknown-transaction"/>
3968                         <enumeration value="unsupported-qualifier"/>
3969                         <enumeration value="wrong-state"/>
3970                     </restriction>
3971                 </simpleType>

```

```

3972         </element>
3973         <element name="fault-data" type="anyType" minOccurs="0"/>
3974         <element ref="btp:qualifiers" minOccurs="0"/>
3975     </sequence>
3976     <attribute name="id" type="ID"/>
3977 </complexType>
3978 </element>
3979
3980 <element name="enrol" substitutionGroup="btp:message">
3981     <complexType>
3982         <sequence>
3983             <element name="target-additional-information"
3984 type="btp:additional-information" minOccurs="0"/>
3985             <element name="superior-identifier" type="btp:identifier"/>
3986             <element name="reply-requested" type="boolean"/>
3987             <element name="reply-address" type="btp:address"
3988 minOccurs="0"/>
3989             <element name="inferior-address" type="btp:address"
3990 minOccurs="1" maxOccurs="unbounded"/>
3991             <element name="inferior-identifier" type="btp:identifier"/>
3992             <element ref="btp:qualifiers" minOccurs="0"/>
3993         </sequence>
3994         <attribute name="id" type="ID"/>
3995     </complexType>
3996 </element>
3997
3998
3999 <element name="enrolled" substitutionGroup="btp:message">
4000     <complexType>
4001         <sequence>
4002             <element name="target-additional-information"
4003 type="btp:additional-information" minOccurs="0"/>
4004             <element name="inferior-identifier" type="btp:identifier"/>
4005             <element ref="btp:qualifiers" minOccurs="0"/>
4006         </sequence>
4007         <attribute name="id" type="ID"/>
4008     </complexType>
4009 </element>
4010
4011 <element name="resign" substitutionGroup="btp:message">
4012     <complexType>
4013         <sequence>
4014             <element name="target-additional-information"
4015 type="btp:additional-information" minOccurs="0"/>
4016             <element name="superior-identifier" type="btp:identifier"/>
4017             <element name="inferior-identifier" type="btp:identifier"/>
4018             <element name="response-requested" type="boolean"/>
4019             <element ref="btp:qualifiers" minOccurs="0"/>
4020         </sequence>
4021         <attribute name="id" type="ID"/>
4022     </complexType>
4023 </element>
4024

```

```

4025     <element name="resigned" substitutionGroup="btp:message">
4026         <complexType>
4027             <sequence>
4028                 <element name="target-additional-information"
4029 type="btp:additional-information" minOccurs="0"/>
4030                 <element name="inferior-identifier" type="btp:identifier"/>
4031                 <element ref="btp:qualifiers" minOccurs="0"/>
4032             </sequence>
4033             <attribute name="id" type="ID"/>
4034         </complexType>
4035     </element>
4036
4037     <element name="prepare" substitutionGroup="btp:message">
4038         <complexType>
4039             <sequence>
4040                 <element name="target-additional-information"
4041 type="btp:additional-information" minOccurs="0"/>
4042                 <element name="inferior-identifier" type="btp:identifier"/>
4043                 <element ref="btp:qualifiers" minOccurs="0"/>
4044             </sequence>
4045             <attribute name="id" type="ID"/>
4046         </complexType>
4047     </element>
4048
4049     <element name="prepared" substitutionGroup="btp:message">
4050         <complexType>
4051             <sequence>
4052                 <element name="target-additional-information"
4053 type="btp:additional-information" minOccurs="0"/>
4054                 <element name="superior-identifier" type="btp:identifier"/>
4055                 <element name="inferior-identifier" type="btp:identifier"/>
4056                 <element name="default-is-cancel" type="boolean"/>
4057                 <element ref="btp:qualifiers" minOccurs="0"/>
4058             </sequence>
4059             <attribute name="id" type="ID"/>
4060         </complexType>
4061     </element>
4062
4063     <element name="confirm" substitutionGroup="btp:message">
4064         <complexType>
4065             <sequence>
4066                 <element name="target-additional-information"
4067 type="btp:additional-information" minOccurs="0"/>
4068                 <element name="inferior-identifier" type="btp:identifier"/>
4069                 <element ref="btp:qualifiers" minOccurs="0"/>
4070             </sequence>
4071             <attribute name="id" type="ID"/>
4072         </complexType>
4073     </element>
4074
4075     <element name="confirmed" substitutionGroup="btp:message">
4076         <complexType>
4077             <sequence>

```

```

4078         <element name="target-additional-information"
4079 type="btp:additional-information" minOccurs="0"/>
4080         <element name="superior-identifier" type="btp:identifier"/>
4081         <element name="inferior-identifier" type="btp:identifier"/>
4082         <element name="confirmed-received" type="boolean"/>
4083         <element ref="btp:qualifiers" minOccurs="0"/>
4084     </sequence>
4085     <attribute name="id" type="ID"/>
4086 </complexType>
4087 </element>
4088
4089 <element name="cancel" substitutionGroup="btp:message">
4090 <complexType>
4091 <sequence>
4092     <element name="target-additional-information"
4093 type="btp:additional-information" minOccurs="0"/>
4094     <element name="inferior-identifier" type="btp:identifier"/>
4095     <element name="reply-address" type="btp:address"
4096 minOccurs="0"/>
4097     <element ref="btp:qualifiers" minOccurs="0"/>
4098 </sequence>
4099 <attribute name="id" type="ID"/>
4100 </complexType>
4101 </element>
4102
4103 <element name="cancelled" substitutionGroup="btp:message">
4104 <complexType>
4105 <sequence>
4106     <element name="target-additional-information"
4107 type="btp:additional-information" minOccurs="0"/>
4108     <element name="superior-identifier" type="btp:identifier"/>
4109     <element name="inferior-identifier" type="btp:identifier"
4110 minOccurs="0"/>
4111     <element ref="btp:qualifiers" minOccurs="0"/>
4112 </sequence>
4113 <attribute name="id" type="ID"/>
4114 </complexType>
4115 </element>
4116
4117 <element name="confirm-one-phase" substitutionGroup="btp:message">
4118 <complexType>
4119 <sequence>
4120     <element name="target-additional-information"
4121 type="btp:additional-information" minOccurs="0"/>
4122     <element name="inferior-identifier" type="btp:identifier"/>
4123     <element name="report-hazard" type="boolean"/>
4124     <element ref="btp:qualifiers" minOccurs="0"/>
4125 </sequence>
4126 <attribute name="id" type="ID"/>
4127 </complexType>
4128 </element>
4129
4130 <element name="hazard" substitutionGroup="btp:message">

```

```

4131     <complexType>
4132         <sequence>
4133             <element name="target-additional-information"
4134 type="btp:additional-information" minOccurs="0"/>
4135             <element name="superior-identifier" type="btp:identifier"/>
4136             <element name="inferior-identifier" type="btp:identifier"/>
4137             <element name="level">
4138                 <simpleType>
4139                     <restriction base="string">
4140                         <enumeration value="mixed"/>
4141                         <enumeration value="possible"/>
4142                     </restriction>
4143                 </simpleType>
4144             </element>
4145             <element ref="btp:qualifiers" minOccurs="0"/>
4146         </sequence>
4147         <attribute name="id" type="ID"/>
4148     </complexType>
4149 </element>
4150
4151     <element name="contradiction" substitutionGroup="btp:message">
4152         <complexType>
4153             <sequence>
4154                 <element name="target-additional-information"
4155 type="btp:additional-information" minOccurs="0"/>
4156                 <element name="inferior-identifier" type="btp:identifier"/>
4157                 <element ref="btp:qualifiers" minOccurs="0"/>
4158             </sequence>
4159             <attribute name="id" type="ID"/>
4160         </complexType>
4161 </element>
4162
4163     <element name="superior-state" substitutionGroup="btp:message">
4164         <complexType>
4165             <sequence>
4166                 <element name="target-additional-information"
4167 type="btp:additional-information" minOccurs="0"/>
4168                 <element name="inferior-identifier" type="btp:identifier"/>
4169                 <element name="status">
4170                     <simpleType>
4171                         <restriction base="string">
4172                             <enumeration value="active"/>
4173                             <enumeration value="prepared-received"/>
4174                             <enumeration value="inaccessible"/>
4175                             <enumeration value="unknown"/>
4176                         </restriction>
4177                     </simpleType>
4178                 </element>
4179                 <element name="reply-requested" type="boolean"/>
4180                 <element ref="btp:qualifiers" minOccurs="0"/>
4181             </sequence>
4182             <attribute name="id" type="ID"/>
4183         </complexType>

```

```

4184     </element>
4185
4186     <element name="inferior-state" substitutionGroup="btp:message">
4187         <complexType>
4188             <sequence>
4189                 <element name="target-additional-information"
4190 type="btp:additional-information" minOccurs="0"/>
4191                 <element name="superior-identifier" type="btp:identifier"/>
4192                 <element name="inferior-identifier" type="btp:identifier"/>
4193                 <element name="status">
4194                     <simpleType>
4195                         <restriction base="string">
4196                             <enumeration value="active"/>
4197                             <enumeration value="inaccessible"/>
4198                             <enumeration value="unknown"/>
4199                         </restriction>
4200                     </simpleType>
4201                 </element>
4202                 <element name="reply-requested" type="boolean"/>
4203                 <element ref="btp:qualifiers" minOccurs="0"/>
4204             </sequence>
4205             <attribute name="id" type="ID"/>
4206         </complexType>
4207     </element>
4208
4209     <element name="redirect" substitutionGroup="btp:message">
4210         <complexType>
4211             <sequence>
4212                 <element name="target-additional-information"
4213 type="btp:additional-information" minOccurs="0"/>
4214                 <element name="superior-identifier" type="btp:identifier"
4215 minOccurs="0"/>
4216                 <element name="inferior-identifier" type="btp:identifier"
4217 />
4218                 <element name="old-address" type="btp:address"
4219 maxOccurs="unbounded"/>
4220                 <element name="new-address" type="btp:address"
4221 maxOccurs="unbounded"/>
4222                 <element ref="btp:qualifiers" minOccurs="0"/>
4223             </sequence>
4224             <attribute name="id" type="ID"/>
4225         </complexType>
4226     </element>
4227
4228
4229     <element name="begin" substitutionGroup="btp:message">
4230         <complexType>
4231             <sequence>
4232                 <element name="target-additional-information"
4233 type="btp:additional-information" minOccurs="0"/>
4234                 <element name="reply-address" type="btp:address"
4235 minOccurs="0"/>
4236                 <element name="transaction-type" type="btp:superior-type"/>

```

```

4237         <element ref="btp:qualifiers" minOccurs="0"/>
4238     </sequence>
4239     <attribute name="id" type="ID"/>
4240 </complexType>
4241 </element>
4242
4243     <element name="begun" substitutionGroup="btp:message">
4244         <complexType>
4245             <sequence>
4246                 <element name="target-additional-information"
4247 type="btp:additional-information" minOccurs="0"/>
4248                 <element name="decider-address" type="btp:address"
4249 minOccurs="0" maxOccurs="unbounded"/>
4250                 <element name="transaction-identifier"
4251 type="btp:identifier" minOccurs="0"/>
4252                 <element name="inferior-handle" type="btp:identifier"
4253 minOccurs="0"/>
4254                 <element name="inferior-address" type="btp:address"
4255 minOccurs="0" maxOccurs="unbounded"/>
4256                 <element ref="btp:qualifiers" minOccurs="0"/>
4257             </sequence>
4258             <attribute name="id" type="ID"/>
4259         </complexType>
4260 </element>
4261
4262     <element name="prepare-inferiors" substitutionGroup="btp:message">
4263         <complexType>
4264             <sequence>
4265                 <element name="target-additional-information"
4266 type="btp:additional-information" minOccurs="0"/>
4267                 <element name="reply-address" type="btp:address"
4268 minOccurs="0"/>
4269                 <element name="transaction-identifier"
4270 type="btp:identifier"/>
4271                 <element name="inferiors-list" minOccurs="0">
4272                     <complexType>
4273                         <sequence>
4274                             <element name="inferior-handle"
4275 type="btp:identifier" maxOccurs="unbounded"/>
4276                         </sequence>
4277                     </complexType>
4278                 </element>
4279                 <element ref="btp:qualifiers" minOccurs="0"/>
4280             </sequence>
4281             <attribute name="id" type="ID"/>
4282         </complexType>
4283 </element>
4284
4285     <element name="confirm-transaction" substitutionGroup="btp:message">
4286         <complexType>
4287             <sequence>
4288                 <element name="target-additional-information"
4289 type="btp:additional-information" minOccurs="0"/>

```

```

4290         <element name="reply-address" type="btp:address"
4291 minOccurs="0"/>
4292         <element name="transaction-identifier"
4293 type="btp:identifier"/>
4294         <element name="inferiors-list" minOccurs="0">
4295             <complexType>
4296                 <sequence>
4297                     <element name="inferior-handle"
4298 type="btp:identifier" maxOccurs="unbounded"/>
4299                 </sequence>
4300             </complexType>
4301         </element>
4302         <element name="report-hazard" type="boolean"/>
4303         <element ref="btp:qualifiers" minOccurs="0"/>
4304     </sequence>
4305     <attribute name="id" type="ID"/>
4306 </complexType>
4307 </element>
4308
4309     <element name="transaction-confirmed" substitutionGroup="btp:message">
4310         <complexType>
4311             <sequence>
4312                 <element name="target-additional-information"
4313 type="btp:additional-information" minOccurs="0"/>
4314                 <element name="transaction-identifier"
4315 type="btp:identifier"/>
4316                 <element ref="btp:qualifiers" minOccurs="0"/>
4317             </sequence>
4318             <attribute name="id" type="ID"/>
4319         </complexType>
4320     </element>
4321
4322     <element name="cancel-transaction" substitutionGroup="btp:message">
4323         <complexType>
4324             <sequence>
4325                 <element name="target-additional-information"
4326 type="btp:additional-information" minOccurs="0"/>
4327                 <element name="reply-address" type="btp:address"
4328 minOccurs="0"/>
4329                 <element name="transaction-identifier"
4330 type="btp:identifier"/>
4331                 <element name="report-hazard" type="boolean"/>
4332                 <element ref="btp:qualifiers" minOccurs="0"/>
4333             </sequence>
4334             <attribute name="id" type="ID"/>
4335         </complexType>
4336     </element>
4337
4338     <element name="cancel-inferiors" substitutionGroup="btp:message">
4339         <complexType>
4340             <sequence>
4341                 <element name="target-additional-information"
4342 type="btp:additional-information" minOccurs="0"/>

```

```

4343         <element name="reply-address" type="btp:address"
4344 minOccurs="0"/>
4345         <element name="transaction-identifier"
4346 type="btp:identifier" minOccurs="0"/>
4347         <element name="inferiors-list">
4348             <complexType>
4349                 <sequence>
4350                     <element name="inferior-handle"
4351 type="btp:identifier" maxOccurs="unbounded"/>
4352                 </sequence>
4353             </complexType>
4354         </element>
4355         <element ref="btp:qualifiers" minOccurs="0"/>
4356     </sequence>
4357     <attribute name="id" type="ID"/>
4358 </complexType>
4359 </element>
4360
4361     <element name="transaction-cancelled" substitutionGroup="btp:message">
4362         <complexType>
4363             <sequence>
4364                 <element name="target-additional-information"
4365 type="btp:additional-information" minOccurs="0"/>
4366                 <element name="transaction-identifier"
4367 type="btp:identifier"/>
4368                 <element ref="btp:qualifiers" minOccurs="0"/>
4369             </sequence>
4370             <attribute name="id" type="ID"/>
4371         </complexType>
4372     </element>
4373
4374     <element name="request-inferior-statuses"
4375 substitutionGroup="btp:message">
4376         <complexType>
4377             <sequence>
4378                 <element name="target-additional-information"
4379 type="btp:additional-information" minOccurs="0"/>
4380                 <element name="reply-address" type="btp:address"
4381 minOccurs="0"/>
4382                 <element name="target-identifier" type="btp:identifier"/>
4383                 <element name="inferiors-list" minOccurs="0">
4384                     <complexType>
4385                         <sequence>
4386                             <element name="inferior-handle"
4387 type="btp:identifier" maxOccurs="unbounded"/>
4388                         </sequence>
4389                     </complexType>
4390                 </element>
4391                 <element ref="btp:qualifiers" minOccurs="0"/>
4392             </sequence>
4393             <attribute name="id" type="ID"/>
4394         </complexType>
4395     </element>

```

```

4396
4397     <element name="inferior-statuses" substitutionGroup="btp:message">
4398         <complexType>
4399             <sequence>
4400                 <element name="target-additional-information"
4401 type="btp:additional-information" minOccurs="0"/>
4402                 <element name="responders-identifier"
4403 type="btp:identifier"/>
4404                 <element name="status-list">
4405                     <complexType>
4406                         <sequence>
4407                             <element name="status-item" maxOccurs="unbounded">
4408                                 <complexType>
4409                                     <sequence>
4410                                         <element name="inferior-handle"
4411 type="btp:identifier"/>
4412                                         <element name="status">
4413                                             <simpleType>
4414                                                 <restriction base="string">
4415                                                     <enumeration value="active"/>
4416                                                     <enumeration value="resigned"/>
4417                                                     <enumeration value="preparing"/>
4418                                                     <enumeration value="prepared"/>
4419                                                     <enumeration value="autonomously-confirmed"/>
4420                                                     <enumeration value="autonomously-cancelled"/>
4421                                                     <enumeration value="confirming"/>
4422                                                     <enumeration value="confirmed"/>
4423                                                     <enumeration value="cancelling"/>
4424                                                     <enumeration value="cancelled"/>
4425                                                     <enumeration value="cancel-contradiction"/>
4426                                                     <enumeration value="confirm-contradiction"/>
4427                                                     <enumeration value="hazard"/>
4428                                                     <enumeration value="invalid"/>
4429                                                 </restriction>
4430                                             </simpleType>
4431                                         </element>
4432                                         <element ref="btp:qualifiers" minOccurs="0"/>
4433                                     </sequence>
4434                                 </complexType>
4435                             </element>
4436                         </sequence>
4437                     </complexType>
4438                 </element>
4439                 <element ref="btp:qualifiers" minOccurs="0"/>
4440             </sequence>
4441             <attribute name="id" type="ID"/>
4442         </complexType>
4443     </element>
4444
4445
4446 </schema>
4447

```

XML schema for standard qualifiers

4448
4449
4450
4451
4452
4453
4454
4455
4456
4457
4458
4459
4460
4461
4462
4463
4464
4465
4466
4467
4468
4469
4470
4471
4472
4473
4474
4475
4476
4477
4478
4479
4480
4481
4482
4483
4484
4485
4486
4487
4488
4489
4490
4491
4492
4493
4494
4495
4496
4497
4498
4499

```
<?xml version="1.0"?>
<schema
  xmlns="http://www.w3.org/2001/XMLSchema"
  targetNamespace="urn:oasis:names:tc:BTP:qualifiers"
  xmlns:btpq="urn:oasis:names:tc:BTP:qualifiers"
  xmlns:btp="urn:oasis:names:tc:BTP:xml"
  elementFormDefault="qualified">

  <element name="transaction-timelimit"
substitutionGroup="btp:qualifier">
    <complexType>
      <complexContent>
        <extension base="btp:qualifier-type">
          <sequence>
            <element name="timelimit"
type="nonNegativeInteger"/>
          </sequence>
        </extension>
      </complexContent>
    </complexType>
  </element>

  <element name="inferior-timeout" substitutionGroup="btp:qualifier">
    <complexType>
      <complexContent>
        <extension base="btp:qualifier-type">
          <sequence>
            <element name="timelimit"
type="nonNegativeInteger"/>
            <element name="intended-decision">
              <simpleType>
                <restriction base="string">
                  <enumeration value="confirm"/>
                  <enumeration value="cancel"/>
                </restriction>
              </simpleType>
            </element>
          </sequence>
        </extension>
      </complexContent>
    </complexType>
  </element>

  <element name="minimum-inferior-timeout"
substitutionGroup="btp:qualifier">
    <complexType>
      <complexContent>
        <extension base="btp:qualifier-type">
          <sequence>
```

```
4500         <element name="minimum-timeout"
4501 type="nonNegativeInteger" />
4502         </sequence>
4503     </extension>
4504 </complexContent>
4505 </complexType>
4506 </element>
4507
4508 <element name="inferior-name" substitutionGroup="btp:qualifier">
4509     <complexType>
4510         <complexContent>
4511             <extension base="btp:qualifier-type">
4512                 <sequence>
4513                     <element name="inferior-name" type="string" />
4514                 </sequence>
4515             </extension>
4516         </complexContent>
4517     </complexType>
4518 </element>
4519
4520 </schema>
4521
```

4521
4522

4523 **Carrier Protocol Bindings**

4524

4525 The notion of bindings is introduced to act as the glue between the BTP messages and an
4526 underlying transport. A binding specification must define various particulars of how the BTP
4527 messages are carried and some aspects of how the related application messages are carried.
4528 This document specifies two bindings: a SOAP binding and a SOAP + Attachments binding.
4529 However, other bindings could be specified by the Oasis BTP technical committee or by a
4530 third party. For example, in the future a binding might exist to put a BTP message directly on
4531 top of HTTP without the use of SOAP, or a closed community could define their own
4532 binding. To ensure that such specifications are complete, the Binding Proforma defines the
4533 information that must be included in a binding specification.
4534

4535 **Carrier Protocol Binding Proforma**

4536

4537 A BTP carrier binding specification should provide the following information:
4538

4539 **Binding name:** A name for the binding, as used in the “binding name” field of BTP
4540 addresses (and available for declaring the capabilities of an implementation). Binding
4541 specified in this document, and future revisions of this document have binding names that are
4542 simple strings of letters, numbers and hyphens (and, in particular, do not contain colons).
4543 Bindings specified elsewhere shall have binding names that are URIs. Bindings specified in
4544 this document use numbers to identify the version of the binding, not the version(s) of the
4545 carrier protocol.
4546

4547 **Binding address format:** This section states the format of the “binding address” field of a
4548 BTP address for this binding. For many bindings, this will be a URL of some kind; for other
4549 bindings it may be some other form
4550

4551 **BTP message representation:** This section will define how BTP messages are represented.
4552 For many bindings, the BTP message syntax will be as specified in the XML schema defined
4553 in this document, and the normal string encoding of that XML will be used.
4554

4555 **Mapping for BTP messages (unrelated) :** This section will define how BTP messages that
4556 are not related to application messages are sent in either direction between Superior and
4557 Inferior. (i.e. those messages sent directly between BTP actors). This mapping need not be
4558 symmetric (i.e. Superior to Inferior may differ to some degree to Inferior to Superior). The
4559 mapping may define particular rules for particular BTP messages, or messages with particular
4560 parameter values (e.g. the FAULT message with “fault-type” “CommunicationFailure” will
4561 typically not be sent as a BTP message). The mapping states any constraints or requirements
4562 on which BTP may or must be bundled together by compounding.
4563

4564 **Mapping for BTP messages related to application messages:** This section will define how
4565 BTP messages that are related to application messages are sent. A binding specification may
4566 defer details of this to a particular application (e.g. a mapping specification could just say

4567 “the CONTEXT may be carried as a parameter of an application invocation”). Alternatively,
4568 the binding may specify a general method that represents the relationship between application
4569 and BTP messages.

4570
4571 **Implicit messages:** This section specifies which BTP messages, if any, are not sent explicitly
4572 but are treated as implicit in application messages or other BTP messages. This may depend
4573 on particular parameter values of the BTP messages or the application messages.

4574
4575 **Faults:** The relationship between the fault and exception reporting mechanisms of the carrier
4576 protocol and of BTP shall be defined. This may include definition of which carrier protocol
4577 exceptions are equivalent to a FAULT/communication-failure message.

4578
4579 **Relationship to other bindings:** Any relationship to other bindings is defined in this section.
4580 If BTP addresses with different bindings are be considered to match (for purposes of
4581 identifying the peer Superior/Inferior and redirection), this should be specified here.

4582
4583 **Limitations on BTP use:** Any limitations on the full range of BTP functionality that are
4584 imposed by use of this binding should be listed. This would include limitations on which
4585 messages can be sent, which event sequences are supported and restrictions on parameter
4586 values. Such limitations may reduce the usefulness of an implementation, but may be
4587 appropriate in certain environments.

4588
4589 **Other:** Other features of the binding, especially any that will potentially affect interoperation
4590 should be specified here. This may include restrictions or requirements on the use or support
4591 of optional carrier parameters or mechanisms.

4592 4593 **Bindings for request/response carrier protocols**

4594
4595 BTP does not generally follow request/response pattern. In particular, on the outcome
4596 relationship either side may initiate a message – this is an essential part of the presume-abort
4597 recovery paradigm although it is not limited to recovery cases. However, there are some BTP
4598 messages, especially in the control relationship, that do have a request/response pattern.
4599 Many (potential) carrier protocols (e.g. HTTP) do have a request/response pattern. The
4600 specification of a binding specification to a request/response carrier protocol needs to state
4601 what rules apply – which messages can be carried by requests, which by responses. The
4602 simplest rule is to send all BTP messages on requests, and let the carrier responses travel back
4603 empty. This would be inefficient in use of network resources, and possibly inconvenient
4604 when used for the BTP request/response pairs.

4605
4606 This section defines a set of rules that allow more efficient use of the carrier, while allowing
4607 the initiator of a BTP request/response pair to ensure the BTP response is sent back on the
4608 carrier response. These rules are specified in this section to enable binding specifications to
4609 reference them, without requiring each binding specification to repeat similar information.

4610
4611 A binding to a request/response carrier is not required to use these rules. It may define other
4612 rules.

4613

4614
4615
4616
4617
4618
4619
4620
4621
4622
4623
4624
4625
4626
4627
4628
4629
4630
4631
4632
4633
4634
4635
4636
4637
4638
4639
4640
4641
4642
4643
4644
4645
4646
4647
4648
4649
4650
4651
4652
4653
4654
4655
4656
4657
4658
4659

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 `btpr:messages` and transmit this `btpr: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 `btpr: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 `btpr: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 `btpr:messages` element, as a bundle of one element.

- 4660 e) A BTP actor that receives a carrier protocol request carrying BTP messages that
4661 do have a reply address, or which initiate processing that produces BTP messages
4662 whose target binding address matches the origin of the request, **may** freely
4663 choose whether to use the carrier protocol response for the replies, or to send
4664 back an “empty carrier protocol response”, and send the BTP replies in a
4665 separately initiated carrier protocol request. The characteristics of an “empty
4666 carrier protocol response” shall be stated in the particular binding specification.
4667
- 4668 f) A BTP actor that sends BTP messages on a carrier protocol request **must** be able
4669 to accept returning BTP messages on the corresponding carrier protocol response
4670 and, if the actor has offered an address on which it will receive carrier requests,
4671 must be able to accept “replying” BTP messages on a separate carrier protocol
4672 request.
4673

4674 SOAP Binding

4675
4676 This binding describes how BTP messages will be carried using SOAP as in the [SOAP 1.1](#)
4677 specification, using the SOAP literal messaging style conventions. If no application message
4678 is sent at the same time, the BTP messages are contained within the SOAP Body element. If
4679 application messages are sent, the BTP messages are contained in the SOAP Header element.
4680

4681 **Binding name:** soap-http-1

4682
4683 **Binding address format:** shall be a URL, of type HTTP.
4684

4685 **BTP message representation:** The string representation of the XML, as specified in the
4686 XML schema defined in this document shall be used. The BTP XML messages are embedded
4687 in the SOAP message without the use of any specific encoding rules (literal style SOAP
4688 message); hence the encodingStyle attribute need not be set or can be set to an empty string.
4689

4690 **Mapping for BTP messages (unrelated):** The “request/response exploitation” rules shall be
4691 used.
4692

4693 BTP messages sent on an HTTP request or HTTP response which is not carrying an
4694 application message, the messages are contained in a single btp:messages element which is
4695 the immediate child element of the SOAP Body element.
4696

4697 An “empty carrier protocol response” sent after receiving an HTTP request containing a
4698 btp:messages element in the SOAP Body and the implementation BTP actor chooses just to
4699 reply at the lower level (and when the request/response exploitation rules allow an empty
4700 carrier protocol response), shall be any of:

- 4701 a) an empty HTTP response
- 4702 b) an HTTP response containing an empty SOAP Envelope
- 4703 c) an HTTP response containing a SOAP Envelope containing a single, empty
4704 btp:messages element.
4705

4706 The receiver (the initial sender of the HTTP request) shall treat these in the same way – they
4707 have no effect on the BTP sequence (other than indicating that the earlier sending did not
4708 cause a communication failure.)

4709
4710
4711

4712 If an application message is being sent at the same time, the mapping for related messages
4713 shall be used, as if the BTP messages were related to the application message. (There is no
4714 ambiguity in whether the BTP messages are related, because only CONTEXT and ENROL
4715 can be related to an application message.)

4716

4717 **Mapping for BTP messages related to application messages:** All BTP messages sent with
4718 an application message, whether related to the application message or not, shall be sent in a
4719 single btp:messages element in the SOAP Header. There shall be precisely one btp:messages
4720 element in the SOAP Header.

4721

4722 The “request/response exploitation” rules shall apply to the BTP messages carried in the
4723 SOAP Header, as if they had been carried in a SOAP Body, unrelated to an application
4724 message, sent to the same binding address.

4725
4726
4727

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.

4728
4729
4730
4731
4732

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.

4733
4734
4735

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.

4736
4737

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

4738
4739
4740
4741
4742
4743

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

4744
4745
4746

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

4747 **Relationship to other bindings:** A BTP address for Superior or Inferior that has the binding
4748 string “soap-http-1” is considered to match one that has the binding string “soap-attachments-
4749 http-1” if the binding address and additional information fields match.

4750
4751 **Limitations on BTP use:** None

4752
4753 **Other:** The SOAP BTP binding does not make use of SOAPAction HTTP header or actor
4754 attribute. The SOAPAction HTTP header is left to be application specific when there are
4755 application messages in the SOAP Body, as an already existing web service that is being
4756 upgraded to use BTP might have already made use of SOAPAction. The SOAPAction HTTP
4757 header shall be omitted when the SOAP message carries only BTP messages in the SOAP
4758 Body.

4759
4760 The SOAP mustUnderstand attribute, when used on the btp:messages containing a BTP
4761 CONTEXT, ensures that the receiver (server, as a whole) supports BTP sufficiently to
4762 determine whether any enrolments are necessary and replies with CONTEXT_REPLY as
4763 appropriate. The sender of the CONTEXT (and related application message) can use this to
4764 ensure that the application work is performed as part of the business transaction, assuming the
4765 receiver’s SOAP implementation supports the mustUnderstand attribute. If mustUnderstand if
4766 false, a receiver can ignore the CONTEXT (if BTP is not supported there), and no
4767 CONTEXT_REPLY will be returned. It is a local option on the sender (client) side whether
4768 the absence of a CONTEXT_REPLY is assumed to be equivalent to aCONTEXT_REPLY/ok
4769 (and the business transaction allowed to proceed to confirmation).

4770
4771 Note – some SOAP implementations may not support the mustUnderstand attribute sufficiently to
4772 enforce these requirements.

4773 **Example scenario using SOAP binding**

4774
4775 The example below shows an application request with CONTEXT message sent from
4776 client.example.com (which includes the Superior) to services.example.com (Service).

```
4777  
4778  
4779 <soap:Envelope  
4780     xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"  
4781     soap:encodingStyle="">  
4782  
4783     <soap:Header>  
4784  
4785         <btp:messages xmlns:btp="urn:oasis:names:tc:BTP:xml">  
4786             <btp:context superior-type="atom">  
4787                 <btp:superior-address>  
4788                     <btp:binding>soap-http-1</btp:binding>  
4789                     <btp:binding-  
4790 address>http://client.example.com/soaphandler</btp:binding-  
4791 address>  
4792                     <btp:additional-information>btpengine</btp:additional-  
4793 information>  
4794                     </btp:superior-address>
```

```

4795         <btp:superior-
4796 identifier>http://example.com/1001</btp:superior-identifier>
4797         <btp:qualifiers>
4798             <btpq:transaction-timelimit
4799 xmlns:btpq="urn:oasis:names:tc:BTP:qualifiers"><btpq:timelimit>180
4800 0</btpq:timelimit></btpq:transaction-timelimit>
4801             </btp:qualifiers>
4802         </btp:context>
4803     </btp:messages>
4804
4805 </soap:Header>
4806
4807 <soap:Body>
4808
4809     <ns1:orderGoods
4810 xmlns:ns1="http://example.com/2001/Services/xyzgoods">
4811         <custID>ABC8329045</custID>
4812         <itemID>224352</itemID>
4813         <quantity>5</quantity>
4814     </ns1:orderGoods>
4815
4816 </soap:Body>
4817
4818 </soap:Envelope>
4819
4820

```

The example below shows CONTEXT_REPLY and a related 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)

```

4828 <soap:Envelope
4829     xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
4830     soap:encodingStyle="">
4831
4832 <soap:Header>
4833 </soap:Header>
4834
4835 <soap:Body>
4836
4837     <btp:messages xmlns:btp="urn:oasis:names:tc:BTP:xml">
4838         <btp:related-group>
4839             <btp:context-reply>
4840                 <btp:target-additional-information>btpengine</btp:target-
4841 additional-information>
4842                 <btp:superior-
4843 identifier>http://example.com/1001</btp:superior-identifier>
4844                 <completion-status>related</completion-status>
4845             </btp:context-reply>
4846

```

```
4847         <btpe:enrol reply-requested="false">
4848         <btpe:target-additional-
4849 information>btpeengine</btpe:target-additional-information>
4850         <btpe:superior-
4851 identifier>http://example.com/1001</btpe:superior-identifier>
4852         <btpe:inferior-address>
4853         <btpe:binding>soap-http-1</btpe:binding>
4854         <btpe:binding-address>
4855         http://services.example.com/soaphandler
4856         </btpe:binding-address>
4857         </btpe:inferior-address>
4858         <btpe:inferior-identifier>
4859         http://example.com/AAAB
4860         </btpe:inferior-identifier>
4861         </btpe:enrol>
4862
4863     </btpe:related-group>
4864
4865 </btpe:messages>
4866
4867 </soap:Body>
4868
4869 </soap:Envelope>
4870
4871
```

4872 SOAP + Attachments Binding

4873
4874 This binding describes how BTP messages will be carried using SOAP as in the [SOAP](#)
4875 [Messages with Attachments](#) specification. It is a superset of the Basic SOAP binding, soap-
4876 http-1. The two bindings only differ when application messages are sent.

4877
4878 **Binding name:** soap-attachments-http-1

4879
4880 **Binding address format:** as for soap-http-1

4881
4882 **BTP message representation:** As for soap-http-1

4883
4884 **Mapping for BTP messages (unrelated):** As for “soap-http-1”, except the SOAP Envelope
4885 containing the SOAP Body containing the BTP messages shall be in a MIME body part, as
4886 specified in [SOAP Messages with Attachments](#) specification. If an application message is
4887 being sent at the same time, the mapping for related messages for this binding shall be used,
4888 as if the BTP messages were related to the application message(s).

4889
4890 **Mapping for BTP messages related to application messages:** MIME packaging shall be
4891 used. One of the MIME multipart/related parts shall contain a SOAP Envelope, whose SOAP
4892 Headers element shall contain precisely one btpe:messages element, containing any BTP
4893 messages. Any BTP CONTEXT in the btpe:messages is considered to be related to the
4894 application message(s) in the SOAP Body, and to also any of the MIME parts referenced
4895 from the SOAP Body (using the “href” attribute).

4896
4897
4898
4899
4900
4901
4902
4903
4904
4905
4906
4907
4908
4909
4910
4911
4912
4913
4914
4915
4916
4917
4918
4919
4920
4921
4922
4923
4924
4925
4926
4927
4928
4929
4930
4931
4932
4933
4934
4935
4936
4937
4938
4939
4940
4941
4942
4943
4944
4945
4946

Implicit messages: As for soap-http-1.

Faults: As for soap-http-1.

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

Limitations on BTP use: None

Other: As for soap-http-1

Example using SOAP + Attachments binding

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

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

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

  <soap:Header>

    <btp:messages xmlns:btp="urn:oasis:names:tc:BTP:xml">
      <btp:context superior-type="atom">
        <btp:superior-address>
          <btp:binding>soap-http-1</btp:binding>
          <btp:binding-address>
            http://client.example.com/soaphandler
          </btp:binding-address>
          </btp:superior-address>
          <btp:superior-
            identifier>http://example.com/1001</btp:superior-identifier>
          </btp:context>
        </btp:messages>

      </soap:Header>

      <soap:Body>
        <orderGoods href="cid:anotherID"/>
      </soap:Body>

    </soap:Envelope>
```

4947
4948
4949
4950
4951
4952
4953
4954
4955
4956
4957
4958
4959
4960
4961
4962

```
--MIME_boundary
Content-Type: text/xml
Content-ID: anotherID

  <ns1:orderGoods
xmlns:ns1="http://example.com/2001/Services/xyzgoods">
  <custID>ABC8329045</custID>
  <itemID>224352</itemID>
  <quantity>5</quantity>
</ns1:orderGoods>

--MIME_boundary--
```

4963 **Conformance**

4964
4965
4966
4967
4968
4969
4970
4971
4972
4973
4974

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.

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
Atomic Hub	Sub-coordinator
	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

4975
4976
4977
4978
4979

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

4980
4981
4982
4983
4984

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

4985 sends ENROL with the same inferior address and with the reply address absent (because the
4986 Inferior in all transactions are dealt with by the same addressable entity), must not assume
4987 that the same is true of received ENROLs)
4988
4989

4989 Part 3. Appendices

4990

4991 *The glossary is the subject of issue 4*

4992

4993 **A. Glossary**

4994

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.
Peer	The other party in a two-party relationship, as in Superior to Inferior, or Sender to Receiver
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. <i>PRF - suffix ? I've used "additional information"</i>
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	<p>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</p> <p>“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”.</p>
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	A set of participants (which may have only one member), all of which will receive instructions that will result in a homogeneous outcome.
<i>or</i>	(Transitively, a set of operations, whose effect is capable of countereffect.)
Atom	An atom is identified by an atom identifier.
Atom Identifier	A globally unique identifier assigned to an atom.
	<p><i>PRF – abs msgs define as unambiguous in scope of its address-as-superior, I think.</i></p>

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.

4995