

2 Business Transaction Protocol

3 An OASIS Committee Specification

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

5 Version 1.0 [0.9.2.3]

6 DD Mmm 2002 [18 March 2002 20:34]

7

8

9

10

<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
<i>Working Draft 0.9.2.1 – issues 2, 3, 15, 19, 50, 67, 95</i>	26 February 2002
<i>Working Draft 0.9.2.2 – as accepted 27 Feb 2002+ corrections, issues 29, 60, 97, 99</i>	12 March 2002
<i>Working Draft 0.9.2.3 – 0.9.2.2 and issue 106, 96, 98</i>	18 March 2002

11 [Change marks relative to 0.9.2.1 with changes accepted](#)

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	28
184	Extensibility.....	29
185	Messages.....	30
186	Qualifiers	30
187	Messages not restricted to outcome or control relationships.	31
188	CONTEXT.....	31
189	CONTEXT_REPLY	32
190	REQUEST_STATUS	33
191	STATUS	34
192	FAULT.....	36
193	REQUEST_INFERIOR_STATUSES, INFERIOR_STATUSES	39
194	Messages used in the outcome relationships	39
195	ENROL	39

196	ENROLLED	41
197	RESIGN	41
198	RESIGNED	42
199	PREPARE	43
200	PREPARED	44
201	CONFIRM	45
202	CONFIRMED	46
203	CANCEL	47
204	CANCELLED	48
205	CONFIRM_ONE_PHASE	49
206	HAZARD	50
207	CONTRADICTION	51
208	SUPERIOR_STATE	52
209	INFERIOR_STATE	54
210	REDIRECT	56
211	Messages used in control relationships	57
212	BEGIN	57
213	BEGUN	58
214	PREPARE_INFERIORS	59
215	CONFIRM_TRANSACTION	60
216	TRANSACTION_CONFIRMED	62
217	CANCEL_TRANSACTION	63
218	CANCEL_INFERIORS	64
219	TRANSACTION_CANCELLED	65
220	REQUEST_INFERIOR_STATUSES	66
221	INFERIOR_STATUSES	67
222	Groups – combinations of related messages	69
223	CONTEXT & application message	69
224	CONTEXT_REPLY & ENROL	70
225	CONTEXT_REPLY (& ENROL) & PREPARED / & CANCELLED	71
226	CONTEXT_REPLY & ENROL & application message (& PREPARED)	72
227	BEGUN & CONTEXT	73
228	BEGIN & CONTEXT	73
229	Standard qualifiers	73
230	Transaction timelimit	73
231	Inferior timeout	74
232	Minimum inferior timeout	75
233	Inferior name	75
234	State Tables	77
235	Explanation of the state tables	77
236	Status queries	77
237	Decision events	77
238	Disruptions – failure events	78
239	Invalid cells and assumptions of the communication mechanism	78
240	Meaning of state table events	79
241	Persistent information	83
242	Failure Recovery	96

243	Types of failure	96
244	Persistent information	97
245	Redirection.....	98
246	Terminator:Decider failures.....	99
247	XML representation of Message Set.....	99
248	Addresses	100
249	Qualifiers	100
250	Identifiers	101
251	Message References.....	101
252	Messages.....	101
253	CONTEXT.....	101
254	CONTEXT_REPLY	101
255	REQUEST_STATUS	102
256	STATUS	102
257	FAULT.....	102
258	ENROL	103
259	ENROLLED	104
260	RESIGN	104
261	RESIGNED.....	104
262	PREPARE	105
263	PREPARED	105
264	CONFIRM	105
265	CONFIRMED	106
266	CANCEL	106
267	CANCELLED.....	106
268	CONFIRM_ONE_PHASE	107
269	HAZARD.....	107
270	CONTRADICTION.....	108
271	SUPERIOR_STATE.....	108
272	INFERIOR_STATE.....	108
273	REDIRECT.....	109
274	BEGIN	109
275	BEGUN.....	109
276	PREPARE_INFERIORS	110
277	CONFIRM_TRANSACTION	110
278	TRANSACTION_CONFIRMED.....	110
279	CANCEL_TRANSACTION	111
280	CANCEL_INFERIORS	111
281	TRANSACTION_CANCELLED.....	112
282	REQUEST_INFERIOR_STATUSES	112
283	INFERIOR_STATUSES	112
284	Standard qualifiers	113
285	Transaction timelimit.....	113
286	Inferior timeout	113
287	Minimum inferior timeout	113
288	Inferior name.....	113
289	Compounding of Messages.....	113

290	XML Schemas	115
291	XML schema for BTP messages.....	115
292	XML schema for standard qualifiers	128
293	Carrier Protocol Bindings	130
294	Carrier Protocol Binding Proforma.....	130
295	Bindings for request/response carrier protocols	131
296	Request/response exploitation rules.....	132
297	SOAP Binding	133
298	Example scenario using SOAP binding	136
299	SOAP + Attachments Binding.....	137
300	Conformance	139
301	Part 3. Appendices.....	142
302	A. Glossary.....	142
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 coordinating publicity for BTP
 - 360 liaising with other standards bodies whose work affects or may be affected by
361 BTP
 - 362 liaising with other standards bodies whose work affects or may be affected by
363 BTP
 - 364 reviewing the appropriate time, in the light of implementation experience and
365 user support, to put BTP forward for adoption as a full OASIS standard
 - 366
 - 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 business-transaction@lists.oasis-open.org
379
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
412 determine whether it is able both to cancel (counter-effect) and to confirm (give final
413 effect to) its effect
- 414
415 report its ability or inability to cancel-or-confirm (its preparedness) to a central
416 coordinating entity

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
755 An Inferior which is also an Atomic Superior.

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 TRANSACTION ~~CONFIRMED~~ ~~COMPLETE~~
808 TRANSACTION ~~CANCELLED~~ ~~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 TRANSACTION_CONFIRMED~~COMPLETE~~,
859 TRANSACTION_CANCELLED~~COMPLETE~~ or (in the case of problems)
860 INFERIOR_STATUSES.

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

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

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

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

881
882 A Terminator receives
883 TRANSACTION_CONFIRMED~~COMPLETE~~
884 TRANSACTION_CANCELLED~~COMPLETE~~
885 INFERIOR_STATUSES

886 887 Initiator

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

892
893 An Initiator sends

894
895 BEGIN
896 BEGIN & CONTEXT

897
898 to a Factory, and receives in reply

899

900 BEGUN & CONTEXT

901

902 **Factory**

903

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

906

907 Decider, which is either

908 Composer or

909 Coordinator

910 Sub-composer

911 Sub-coordinator

912

913 A Factory receives

914

915 BEGIN

916 BEGIN & CONTEXT

917

918 and replies with

919

920 BEGUN & CONTEXT

921

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

925

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

929

930

931

932 **Other roles**

933

934 **Redirector**

935

936 Sends a REDIRECT message to inform a Superior or Inferior ~~by actor~~ that an address
937 previously supplied for the peer (i.e. an Inferior or Superior, respectively) ~~some other actor~~ is
938 no longer appropriate, and to supply a new address or set of addresses to replace the old one.

939

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

942

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

947 REDIRECT message, [in which case the Redirector shall use the new address in the received](#)
948 [message as the target for the REDIRECT that it sends.](#))

949

950 ~~A Redirector may also be used to change the address of other BTP actors.~~

951

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

954

955 **Status Requestor**

956

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

962

963 A Status Requestor sends

964

965 REQUEST_STATUS
966 REQUEST_INFERIOR_STATUSES

967

968 and receives

969

970 STATUS
971 INFERIOR_STATUSES

972

973 in response.

974

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

978

979 **Abstract Messages and Associated Contracts**

980

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

984

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

986

- 987 ❑ deliver messages completely and correctly, or not at all (corrupted messages will
988 not be delivered);
- 989
- 990 ❑ report some communication failures, but will not necessarily report all (i.e. not all
991 message deliveries are positively acknowledged within the carrier);
- 992
- 993 ❑ sometimes deliver successive messages in a different order than they were sent;

994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040

and

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

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

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

Addresses

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

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

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

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

1041 All messages that concern a particular Superior:Inferior relationship have an identifier
1042 parameter for the target side as well as the target address. This allows full flexibility for
1043 implementation choices – an implementation can:
1044
1045 a) Use the same binding address and additional information for multiple business
1046 transactions, using the identifier parameter to locate the relevant state
1047 information;
1048 b) Use the same binding address for multiple business transactions and use the
1049 additional information to locate the information; or
1050 c) Use a different binding address for each business transaction.

1051
1052 Which of these choices is used is opaque to the entity sending the message – both parts of the
1053 address and the identifier originated at the recipient of this message (and were transmitted as
1054 parameters of earlier messages in the opposite direction).

1055
1056 BTP recovery requires that the state information for a Superior or Inferior is accessible after
1057 failure and that the peer can distinguish between temporary inaccessibility and the permanent
1058 non-existence of the state information. As is explained in “Redirection” below, BTP provides
1059 mechanisms – having a set of BTP addresses for some parameters, and the REDIRECT
1060 message – that make this possible, even if the recovered state information is on a different
1061 address to the original one (as may be the case if case c) above is used).

1062

1063

1064 Request/response pairs

1065

1066 Many of the messages combine in pairs as a request and its response. However, in some cases
1067 the response message is sent without a triggering request, or as a possible response to more
1068 than one type of request. To allow for this, the abstract message set treats each message as
1069 standalone; but where a request does expect a reply, a “reply-address” parameter will be
1070 present. For any message with a reply address parameter, in the case of certain errors, a
1071 FAULT message will be sent to the reply address instead of the expected reply.

1072

1073 Between Superior and Inferior the address of the peer is normally known (from the “superior-
1074 address” on an earlier CONTEXT or the “inferior-address” on a received ENROL). However,
1075 in some cases a message will be received for a Superior or Inferior that is not known – the
1076 state information no longer exists. This is not an exceptional condition but occurs when one
1077 side has either not created or has removed its persistent state in accordance with the
1078 procedures, but a message has got lost in a failure, and the peer still has state information.
1079 The response to a message for an unknown (and logically non-existent) Superior is
1080 SUPERIOR_STATE/unknown, for an unknown Inferior it is INFERIOR_STATE/unknown.
1081 However, since the intended target is unknown, there is no information to locate the peer,
1082 which sent the undeliverable message. To enable the receiver to reply with the appropriate
1083 *_STATE/unknown, all the messages between Superior and Inferior have a “senders-
1084 address” parameter. If a FAULT message is to be sent in response to message which (as an
1085 abstract message) has a “senders-address” parameter, the ~~For messages which are specified as~~
1086 ~~sent between Superior and Inferior, a~~ FAULT message is sent to ~~that~~ address-peer.
1087

1088
1089
1090
1091

Note – Both reply-address and senders-address may be absent when the carrier protocol itself has a request/response pattern. In these cases, the reply or sender address is implicitly that of the sender of the request (and thus the destination of a response)

1092
1093

Compounding messages

1094
1095
1096

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

1097
1098
1099
1100
1101
1102
1103

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

1104
1105
1106
1107
1108

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

1109
1110
1111
1112
1113

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

1114
1115
1116
1117
1118
1119
1120
1121

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

1122
1123
1124
1125
1126
1127

A particular and important case of related messages is where a BTP CONTEXT message is sent related to an application message. In this case, the target of the application message defines the destination of the CONTEXT message. The receiving implementation may in fact remove the CONTEXT before delivering the application message to the application (Service) proper, but from the perspective of the sender, the two are sent to the same place.

1128
1129
1130

The compounding mechanisms, and the multi-part address structures, support the “one-wire” and “one-shot” communication patterns.

1131
1132
1133

In “one-wire”, all message exchanges between two sides of a Superior:Inferior relationship, including the associated application messages, pass via the same “endpoints”. These “endpoints” may in fact be relays, routing messages on to particular actors within their

1134 domain. The onward routing will require some further addressing, but this has to be opaque to
1135 the sender. This can be achieved if the relaying endpoint ensures that all addresses for actors
1136 in its domain have the relay's address as their binding address, and any routing information it
1137 will need in its own domain is placed in the additional information. (This may involve the
1138 relay changing addresses in messages as they pass through it on the way out). On receiving a
1139 message, it determines the within-domain destination from the received additional
1140 information (which is thus rewritten) and forwards the message appropriately. The sender is
1141 unaware of this, and merely sees addresses with the same binding address, which it is
1142 permitted to bundle. The content of the "additional information" is a matter only for the relay
1143 – it could put an entire BTP address in there, or other implementation-defined information.
1144 Note that a quite different one-wire implementation can be constructed where there is no
1145 relaying, but the receiving entity effectively performs all roles, using the received identifiers
1146 to locate the appropriate state.

1147
1148 "One-shot" communication makes it possible to send an application message, receive the
1149 application reply, enrol an Inferior to be responsible for the confirm/cancel of the operations
1150 of those message and inform the Superior that the Inferior is prepared, all in one two-way
1151 exchange across the network (e.g. one request/reply of a carrier protocol).. The application
1152 request is sent with a related CONTEXT message. The application response is sent with a
1153 relation group of CONTEXT_REPLY/related, ENROL/no-rsp-req message and a
1154 PREPARED message. This is possible even if the Superior address is different from the
1155 address of the application element that sends the original message (if the application
1156 exchange is request/reply, there may not even be an identifiable address for the application
1157 element). The target addresses of the ENROL and PREPARED (the Superior address) are not
1158 transmitted; the actor that was originally responsible for adding the CONTEXT to the
1159 outbound application message remembers the Superior address and forwards the ENROL and
1160 PREPARED appropriately.

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

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

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

1178 **Extensibility**

1179

1180 To simplify interoperation between implementations of this edition of BTP with
1181 implementations of future editions, the “must-be-understood” sub-parameter as specified for
1182 Qualifiers may be defined for use with any parameter added to an existing message in a future
1183 revision of this specification. The default for “must-be-understood” shall be “true”, so an
1184 implementation receiving an unrecognised parameter without a “false” value for “must-be-
1185 understood” shall not accept it (the FAULT value “UnrecognisedParameter” is available, but
1186 other errors, including lower-layer parsing/unmarshalling errors may be reported instead). If
1187 “must-be-understood” with the value “false” is present as a sub-parameter of a parameter in
1188 any message, a receiving implementation **should** ignore the parameter.

1189
1190 How the sub-parameter is associated with the new parameter is determined by the particular
1191 binding.

1192
1193 No special mechanism is provided to allow for the introduction of completely new messages.
1194

1195 Messages

1196 1197 Qualifiers

1198
1199 All messages have a Qualifiers parameter which contains zero or more Qualifier values. A
1200 Qualifier has sub-parameters:
1201

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

1202
1203 **Qualifier group** ensures the Qualifier name is unambiguous. Qualifiers in the
1204 same group need not have any functional relationship. The qualifier group will
1205 typically be used to identify the specification that defines the qualifier’s meaning
1206 and use. Qualifiers may be defined in this or other standard specifications, in
1207 specifications of a particular community of users or of implementations or by
1208 bilateral agreement.

1209
1210 **Qualifier name** this identifies the meaning and use of the Qualifier, using a name
1211 that is unambiguous within the scope of the Qualifier group.

1212
1213 **Must-be-understood** if this has the value “true” and the receiving entity does
1214 not recognise the Qualifier type (or does not implement the necessary
1215 functionality), a FAULT “UnsupportedQualifier” shall be returned and the
1216 message shall not be processed. Default is “true”.
1217

1218 **To-be-propagated** if this has the value “true” and the receiving entity passes the
 1219 BTP message (which may be a CONTEXT, but can be other messages) onwards
 1220 to other entities, the same Qualifier value shall be included. If the value is
 1221 “false”, the Qualifier shall not be automatically included if the BTP message is
 1222 passed onwards. (If the receiving entity does support the qualifier type, it is
 1223 possible a propagated message may contain another instance of the same type,
 1224 even with the same Content – this is not considered propagation of the original
 1225 qualifier.). Default is “false”.

1226
 1227 **Content** the type (which may be structured) and meaning of the content is
 1228 defined by the specification of the Qualifier.
 1229
 1230

1231 **Messages not restricted to outcome or control relationships.**

1232
 1233 The messages in this section are used between various roles. CONTEXT message is used in
 1234 the Initiator:Factory relationship (when it is related to BEGIN or to BEGUN), and related to
 1235 an application ‘message’ to propagate the business transaction between parts of the
 1236 application. CONTEXT_REPLY is used as the reply to a CONTEXT.REQUEST_STATUS
 1237 can be issued to, and STATUS returned by any of Decider, Superior or Inferior. FAULT can
 1238 be used on any relationship to indicate an error condition back to the sender of a message.
 1239

1240 **CONTEXT**

1241
 1242 A CONTEXT is supplied by (or on behalf of) a Superior and related to one or more
 1243 application messages. (The means by which this relationship is represented is determined by
 1244 the binding and the binding mechanisms of the application protocol.) The “superior-type”
 1245 parameter identifies whether the Superior will apply the same decision to all Inferiors
 1246 enrolled using the same superior identifier (“superior-type” is “atom”) or whether it may
 1247 apply different decisions (“superior-type” is “cohesion”).
 1248

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

1249
 1250
 1251 [superior-address](#)-~~as-superior~~ the address to which ENROL and other
 1252 messages from an enrolled Inferior are to be sent. This can be a set of alternative
 1253 addresses.
 1254

1255 **superior-identifier** identifies the Superior. This shall be globally unambiguous.

1256
1257 **reply-address** the address to which a replying CONTEXT_REPLY is to be sent.
1258 This may be different each time the CONTEXT is transmitted – it refers to the
1259 destination of a replying CONTEXT_REPLY for this particular transmission of
1260 the CONTEXT.

1261
1262 **superior-type** identifies whether the CONTEXT refers to a Cohesion or an
1263 Atom. Default is atom.

1264
1265 **qualifiers** standardised or other qualifiers. The standard qualifier “Transaction
1266 timelimit” is carried by CONTEXT.

1267
1268 There is no “target-address” parameter for CONTEXT as it is only transmitted in relation to
1269 the application messages, BEGIN and BEGUN.

1270
1271 The forms CONTEXT/cohesion and CONTEXT/atom refer to CONTEXT messages with the
1272 “superior-type” with the appropriate value.

1273
1274

CONTEXT_REPLY

1275
1276
1277 CONTEXT_REPLY is sent after receipt of CONTEXT (related to application message(s)) to
1278 indicate whether all necessary enrolments have already completed (ENROLLED has been
1279 received) or will be completed by ENROL messages sent in relation to the
1280 CONTEXT_REPLY or if an enrolment attempt has failed. CONTEXT_REPLY may be sent
1281 related to an application message (typically the response to the application message related to
1282 the CONTEXT). In some bindings the CONTEXT_REPLY may be implicit in the application
1283 message.

1284

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

1285
1286 **target-address** the address to which the CONTEXT_REPLY is sent. This shall
1287 be the “reply-address” from the CONTEXT.

1288
1289 **superior-identifier** the “superior-identifier” from the CONTEXT

1290
1291 **completion-status:** reports whether all enrol operations made necessary by the
1292 receipt of the earlier CONTEXT message have completed. Values are

1293

Value	meaning
-------	---------

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.

1294

1295

qualifiers standardised or other qualifiers.

1296

1297

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

1298

1299

1300

1301

1302

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

1303

1304

1305

1306

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

1307

1308

1309

REQUEST_STATUS

1310

1311

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).

1312

1313

1314

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

1315

1316

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

1317

1318

1319

1320

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

1321

1322

target identifier The identifier for the business transaction, or part of business transaction whose status is sought. If the target-address is an "decider-address~~-as-decider~~", this parameter shall be the "transaction-identifier" on the BEGUN

1323

1324

1325 message. If the “target-address” is an “~~inferior-address-as-inferior~~”, this
 1326 parameter shall be the “inferior-identifier” on the ENROL message. If the
 1327 “target-address” is a an “~~superior-address-as-superior~~”, this parameter shall be
 1328 the “superior-identifier” on the CONTEXT.

1329
 1330 **qualifiers** standardised or other qualifiers.

1331
 1332 Types of FAULT possible (sent to “reply-address”)

- 1333
 1334 *General*
 1335 *Redirect – if the intended target now has a different address*
 1336 *StatusRefused – if the receiver is not prepared to report its status to the*
 1337 *sender of this message*
 1338 *UnknownTransaction – if the target-identifier is unknown*

1339
 1340
 1341 **STATUS**

1342
 1343 Sent by a Inferior, Superior or Decider in reply to a REQUEST_STATUS, reporting the
 1344 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

1346
 1347 **target-address** the address to which the STATUS is sent. This will be the
 1348 “reply-address” on the REQUEST_STATUS message

1349
 1350
 1351 **responders-identifier** the identifier of the state, identical to the “target-
 1352 identifier” on the REQUEST_STATUS.

1353 **status** states the current status of the transaction tree node represented by the
 1354 sender. Some of the values are only issued if the sender is an Inferior. If the
 1355 transaction tree node is both Superior and Inferior (i.e. is a sub-coordinator or
 1356 sub-composer), and two status values would be valid for the current state, it is the
 1357 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

status value	Meaning from Superior	Meaning from Inferior
<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
<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

status value	Meaning from Superior	Meaning from Inferior
<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

1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370

qualifiers standardised or other qualifiers.

Types of FAULT possible

General

FAULT

Sent in reply to various messages to report an error condition . The FAULT message is used on all the relationships as a general negative reply to a message.

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

1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387

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

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

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

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

fault-data information relevant to the particular error. Each “fault-type” defines the content of the “fault-data”:

fault-type	meaning	fault-data
<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 None
<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 "inferior-identifier" in the message or at least one "inferior-identifier"s in an "inferior-list" parameter is not known or does not identify a known Inferior The Superior is known but the Inferior identified by the address-as-inferior and identifier are not enrolled in it	One or more invalid identifiers 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 requested ed status (or inferior statuses) to this StatusRequestor	Free-text explanation None
<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 None
<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 or the transaction identified by related CONTEXT is in an invalid state.	Free-text explanation None
Redirect	The target of the BTP message now has a different address	Set of BTP addresses, to be used instead of the address the BTP message was received on

1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400

UnknownParameter A BTP message has been received with an unrecognised parameter Free text explanation

q
u
fault-text Free text describing the fault or providing more information. Whether this parameter is present, and exactly what it contains are an implementation option.

Equalifiers standardised or other qualifiers.

1401
1402
1403

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

1404
1405
1406

REQUEST_INFERIOR_STATUSES, INFERIOR_STATUSES

1407
1408
1409
1410
1411
1412
1413
1414

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

1415
1416

Messages used in the outcome relationships

1417
1418

ENROL

1419
1420
1421
1422

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

Parameter	type
target-address	BTP address
superior-identifier	Identifier
response-requested	Boolean
reply-address	BTP address
<u>inferior</u> -address-as-inferior	Set of BTP addresses
inferior-identifier	Identifier

qualifiers

List of qualifiers

1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466

target-address the address to which the ENROL is sent. This will be the [“superior-address-as-superior”](#) from the CONTEXT message.

superior-identifier. The “superior-identifier” as on the CONTEXT message

response-requested true if an ENROLLED response is required, false otherwise. Default is false.

reply-address the address to which a replying ENROLLED is to be sent, if “response-requested” is true. If this field is absent and “response-requested” is true, the ENROLLED should be sent to the [“inferior-address-as-inferior”](#) (or one of them, at sender’s option)

[inferior-address-as-inferior](#) the address to which PREPARE, CONFIRM, CANCEL and SUPERIOR_STATE messages for this Inferior are to be sent.

inferior-identifier an identifier that identifies this Inferior. This shall be globally unambiguous..

qualifiers standardised or other qualifiers. The standard qualifier “Inferior name” may be present.

Types of FAULT possible (sent to “reply-address”)

General

InvalidSuperior – if “superior-identifier” is unknown

[Redirect – if the Superior now has a different superior-address-as-superior](#)

DuplicateInferior – if inferior with at least one of the set [“inferior-address-as-inferior”](#) the same and the same “inferior-identifier” is already enrolled

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

The form ENROL/rsp-req refers to an ENROL message with “response-requested” having the value “true”; ENROL/no-rsp-req refers to an ENROL message with “response-requested” having the value “false”

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

1467 **ENROLLED**

1468

1469 Sent from Superior in reply to an ENROL/rsp-req message, to indicate the Inferior has been
1470 successfully enrolled (and will therefore be included in the termination exchanges)

1471

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

1472

1473 **target-address** the address to which the ENROLLED is sent. This will be the
1474 “reply-address” from the ENROL message (or one of the [“inferior-address-as-](#)
1475 [inferior”](#)s if the “reply-address” was empty)

1476

1477 [sender-address](#) the address from which the ENROLLED is sent. This is an
1478 [address of the Superior.](#)

1479

1480 **inferior-identifier** The “inferior-identifier” as on the ENROL message

1481

1482 **qualifiers** standardised or other qualifiers.

1483

1484 No FAULT messages are issued on receiving ENROLLED.

1485

1486

1487 **RESIGN**

1488

1489 Sent from an enrolled Inferior to the Superior to remove the Inferior from the enrolment. This
1490 can only be sent if the operations of the business transaction have had no effect as perceived
1491 by the Inferior.

1492

1493 RESIGN may be sent at any time prior to the sending of a PREPARED or CANCELLED
1494 message (which cannot then be sent). RESIGN may be sent in response to a PREPARE
1495 message.

1496

Parameter	type
target-address	BTP address
sender-address	BTP address
superior-identifier	identifier
inferior-identifier	identifier
response-requested	Boolean

Qualifiers List of qualifiers

1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533

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

[sender-address](#) the address from which the RESIGN is sent. This is an address of the Inferior.

superior-identifier The “superior-identifier” as on the ENROL message

inferior-identifier The “inferior-identifier” as on the earlier ENROL message

response-requested is set to “true” if a RESIGNED response is required. Default is “false”.

qualifiers standardised or other qualifiers.

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

Types of FAULT possible (sent to ~~“inferior”~~sender-address-as inferior”)

General

InvalidSuperior – if “superior-identifier” is unknown

InvalidInferior – if no ENROL had been received for this “inferior-identifier”~~inferior-address-as inferior”~~ and identifier (Inferior Identity)

WrongState – if a PREPARED or CANCELLED has already been received by the Superior from this Inferior

The form RESIGN/rsp-req refers to an RESIGN message with “response-requested” having the value “true”; RESIGN /no-rsp-req refers to an RESIGN message with “response-requested” having the value “false”

RESIGNED

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

Parameter	Type
target-address	BTP address
<u>sender-address</u>	<u>BTP address</u>
inferior-identifier	Identifier
qualifiers	List of qualifiers

1534
 1535 **target-address** the address to which the RESIGNED is sent. This will be the
 1536 “~~inferior-address-as-inferior~~” from the ENROL message.
 1537
 1538 **sender-address** the address from which the RESIGNED is sent. This is an
 1539 address of the Superior.
 1540
 1541 **inferior-identifier** The “inferior-identifier” as on the earlier ENROL message for
 1542 this Inferior.
 1543
 1544 **qualifiers** standardised or other qualifiers.

1545
 1546 After receiving this message the Inferior will not receive any more messages with this
 1547 “~~inferior-address-as-inferior~~” and “~~inferior-identifier~~”.
 1548

1549 Types of FAULT possible (sent to “~~sender-Superior-address~~”)

1550 *General*

1551 *WrongState* - if RESIGN has not been sent

1552
 1553
 1554 **PREPARE**

1555
 1556 Sent from Superior to an Inferior from whom ENROL but neither CANCELLED nor
 1557 RESIGN have been received, requesting a PREPARED message. PREPARE can be sent after
 1558 receiving a PREPARED message.
 1559
 1560

Parameter	Type
target-address	BTP address
<u>sender-address</u>	<u>BTP address</u>
inferior-identifier	Identifier
qualifiers	List of qualifiers

1561
 1562 **target-address** the address to which the PREPARE message is sent. ~~When sent~~
 1563 ~~from Superior to Inferior,~~ this will be the “~~inferior-address-as-inferior~~” from the
 1564 ENROL message.
 1565
 1566 **sender-address** the address from which the PREPARE is sent. This is an
 1567 address of the Superior.
 1568
 1569 **inferior-identifier** ~~When sent from Superior to Inferior,~~ the “inferior-identifier”
 1570 as on the earlier ENROL message.
 1571

1572 **qualifiers** standardised or other qualifiers. The standard qualifier “Minimal
1573 inferior timeout” is carried by PREPARE.

1574
1575
1576 On receiving PREPARE, an Inferior **should** reply with a PREPARED, CANCELLED or
1577 RESIGN.

1578
1579 Types of FAULT possible (sent to ~~Superior~~ “[sender-address](#)”)

1580
1581 **General**
1582 **InvalidInferior** – if “inferior-identifier” is unknown, or an inferior-handle
1583 on the inferiors-list is unknown
1584 **WrongState** – if a CONFIRM or CANCEL has already been received by
1585 this Inferior.

1586
1587
1588 **PREPARED**

1589
1590 Sent from Inferior to Superior, either unsolicited or in response to PREPARE, but only when
1591 the Inferior has determined the operations associated with the Inferior can be confirmed and
1592 can be cancelled, as may be instructed by the Superior. The level of isolation is a local matter
1593 (i.e. it is the Inferiors choice, as constrained by the shared understanding of the application
1594 exchanges) – other access may be blocked, may see applied results of operations or may see
1595 the original state.

1596

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

1597
1598 **target-address** the address to which the PREPARED is sent. This will be the
1599 Superior address as on the ENROL message.

1600
1601 [sender-address](#) the address from which the PREPARED is sent. This is an
1602 address of the Inferior.

1603
1604 **superior-identifier** the “superior-identifier” as on the ENROL message

1605
1606 **inferior-identifier** The “inferior-identifier” as on the ENROL message

1607

1608 **default-is cancel** if “true”, the Inferior states that if the outcome at the Superior
1609 is to cancel the operations associated with this Inferior, no further messages need
1610 be sent to the Inferior. If the Inferior does not receive a CONFIRM message, it
1611 will cancel the associated operations. The value “true” will invariably be used
1612 with a qualifier indicating under what circumstances (usually a timeout) an
1613 autonomous decision to cancel will be made. If “false”, the Inferior will expect
1614 a CONFIRM or CANCEL message as appropriate, even if qualifiers indicate that
1615 an autonomous decision will be made.

1616
1617 **qualifiers** standardised or other qualifiers. The standard qualifier “Inferior
1618 timeout” may be carried by PREPARED.

1619
1620 On sending a PREPARED, the Inferior undertakes to maintain its ability to confirm or cancel
1621 the effects of the associated operations until it receives a CONFIRM or CANCEL message.
1622 Qualifiers may define a time limit or other constraints on this promise. The “default-is
1623 cancel” parameter affects only the subsequent message exchanges and does not of itself state
1624 that cancellation will occur.

1625
1626 Types of FAULT possible (sent to [“inferior:sender-address-as-inferior”](#))

1627
1628 *General*

1629 *InvalidSuperior* – if “superior-identifier” is unknown

1630 *InvalidInferior* – if no ENROL has been received for this [“inferior-](#)
1631 [address-as-inferior”](#) and [“inferior-identifier”](#), or if RESIGN has been
1632 received from this Inferior

1633
1634 The form PREPARED/cancel refers to a PREPARED message with “default-is cancel” =
1635 “true”. The unqualified form PREPARED refers to a PREPARED message with “default-is
1636 cancel” = “false”.

1637
1638

1639 CONFIRM

1640

1641 Sent by the Superior to an Inferior from whom PREPARED has been received.

1642

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

1643

1644 **target-address** the address to which the CONFIRM message is sent. This will
1645 be the [“inferior-address-as-inferior”](#) from the ENROL message.

1646

1647 [sender-address](#) the address from which the CONFIRM is sent. This is an
 1648 [address of the Superior.](#)
 1649
 1650 **inferior-identifier** The “inferior-identifier” as on the earlier ENROL message for
 1651 this Inferior.
 1652
 1653 **qualifiers** standardised or other qualifiers.

1654
 1655 On receiving CONFIRM, the Inferior is released from its promise to be able to undo the
 1656 operations of associated with the Inferior. The effects of the operations can be made available
 1657 to everyone (if they weren’t already).
 1658

1659 Types of FAULT possible (sent to [“sender-Superior-address”](#))

1660
 1661 *General*

1662 *InvalidInferior* – if “inferior-identifier” is unknown

1663 *WrongState* – if no PREPARED has been sent by, or if CANCEL has
 1664 been received by this Inferior.
 1665
 1666

1667 **CONFIRMED**

1668
 1669 Sent after the Inferior has applied the confirmation, both in reply to CONFIRM or when the
 1670 Inferior has made an autonomous confirm decision, and in reply to a
 1671 CONFIRM_ONE_PHASE if the Inferior decides to confirm its associated operations.
 1672
 1673

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

1674
 1675 **target-address** the address to which the CONFIRMED is sent. This will be the
 1676 Superior address as on the CONTEXT message.
 1677

1678 [sender-address](#) the address from which the CONFIRMED is sent. This is an
 1679 [address of the Inferior.](#)
 1680

1681 **superior-identifier** the “superior-identifier” as on the CONTEXT message.
 1682

1683 **inferior-identifier** the “inferior-identifier” as on the earlier ENROL message.
 1684
 1685
 1686 **confirm-received** “true” if CONFIRMED is sent after receiving a CONFIRM
 1687 message; “false” if an autonomous confirm decision has been made and either if
 1688 no CONFIRM message has been received or the implementation cannot
 1689 determine if CONFIRM has been received (due to loss of state information in a
 1690 failure).
 1691
 1692 **qualifiers** standardised or other qualifiers.

1693
 1694 Types of FAULT possible (sent to [“inferior sender-address-as inferior”](#))

1695
 1696 **General**

1697 **InvalidSuperior** – if “superior-identifier” is unknown

1698 **InvalidInferior** – if no ENROL has been received for this [“inferior-
 1699 address-as inferior”](#) and [“inferior-identifier”](#), or if RESIGN has been
 1700 received from this Inferior.
 1701

1702 Note – A CONFIRMED message arriving before a CONFIRM message is
 1703 sent, or after a CANCEL has been sent will occur when the Inferior has
 1704 taken an autonomous decision and is not regarded as occurring in the wrong
 1705 state. (The latter will cause a CONTRADICTION message to be sent.)

1706
 1707 The form CONFIRMED/auto refers to a CONFIRMED message with “confirm-
 1708 received” = “false”; CONFIRMED/response refers to a CONFIRMED message
 1709 with “confirm-received” = “true”.
 1710

1711
 1712 **CANCEL**

1713
 1714 Sent by the Superior to an Inferior at any time before (and unless) CONFIRM has been sent.
 1715

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

1716
 1717 **target-address** the address to which the CANCEL message is sent. This will be
 1718 the [“inferior-address-as inferior”](#) from the ENROL message.
 1719

1720 [sender-address](#) the address from which the CANCEL is sent. This is an address
1721 [of the Superior.](#)

1722
1723 **inferior-identifier** the “inferior-identifier” as on the earlier ENROL message.
1724

1725 **qualifiers** standardised or other qualifiers.
1726

1727 When received by an Inferior, the effects of any operations associated with the Inferior
1728 should be undone. If the Inferior had sent PREPARED, the Inferior is released from its
1729 promise to be able to confirm the operations.
1730

1731 Types of FAULT possible (sent to ~~Superior~~ “[sender-address](#)”)
1732

1733 **General**

1734 **InvalidInferior** – if “inferior-identifier” is unknown, or an inferior-handle
1735 on the inferiors-list is unknown

1736 **WrongState** – if a CONFIRM has been received by this Inferior.
1737

1738
1739

CANCELLED

1740
1741 Sent when the Inferior has applied (or is applying) cancellation of the operations associated
1742 with the Inferior. CANCELLED is sent from Inferior to Superior in the following cases:
1743

- 1744 1. before (and instead of) sending PREPARED, to indicate the Inferior is unable to
1745 apply the operations in full and is cancelling all of them;
1746
1747 2. in reply to CANCEL, regardless of whether PREPARED has been sent;
1748
1749 3. after sending PREPARED and then making and applying an autonomous
1750 decision to cancel.
1751
1752 4. in reply to CONFIRM_ONE_PHASE if the Inferior decides to cancel the
1753 associated operations
1754

1755 As is specified in the state tables, cases 1, 2 and 3 are not always distinct in some
1756 circumstances of recovery and resending of messages.
1757

Parameter

target-address BTP address

[sender-address](#) [BTP address](#)

superior-identifier Identifier

inferior-identifier Identifier

qualifiers List of qualifiers

1758
 1759 **target-address** the address to which the CANCELLED is sent. This will be the
 1760 Superior address as on the CONTEXT message.
 1761
 1762 [sender-address](#) the address from which the CANCELLED is sent. This is an
 1763 [address of the Inferior.](#)
 1764
 1765 **superior-identifier** the “superior-identifier” as on the CONTEXT message.
 1766
 1767 **inferior-identifier** the inferior identifier as on the earlier ENROL message.
 1768
 1769 **qualifiers** standardised or other qualifiers.

1770
 1771 Types of FAULT possible (sent to [“~~inferior~~sender-address-as inferior”](#))

1772
 1773 *General*

1774 *InvalidSuperior* – if “superior-identifier” is unknown

1775 *InvalidInferior* – if no ENROL has been received for this [“~~inferior-~~](#)
 1776 [address-as inferior”](#) and [“inferior-identifier”](#), or if RESIGN has been
 1777 received from this Inferior

1778 *WrongState* – if CONFIRM has been sent
 1779

1780 Note – A CANCELLED message arriving before a CANCEL message is
 1781 sent, or after a CONFIRM has been sent will occur when the Inferior has
 1782 taken an autonomous decision and is not regarded as occurring in the wrong
 1783 state. (The latter will cause a CONTRADICTION message to be sent.)

1784
 1785
 1786 **CONFIRM_ONE_PHASE**

1787
 1788 Sent from a Superior to an enrolled Inferior, when there is only one such enrolled Inferior. In
 1789 this case the two-phase exchange is not performed between the Superior and Inferior and the
 1790 outcome decision for the operations associated with the Inferior is determined by the Inferior.
 1791

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

1792

1793 **target-address** the address to which the CONFIRM_ONE_PHASE message is
1794 sent This will be the “inferior-address-as-inferior” on the ENROL message.

1795
1796 sender-address the address from which the CONFIRM_ONE_PHASE is sent.
1797 This is an address of the Superior.

1798
1799 **inferior-identifier** The “inferior-identifier” as on the earlier ENROL message for
1800 this Inferior.

1801
1802 **report hazard** Defines whether the superior wishes to be informed if a mixed
1803 condition occurs for the operations associated with the Inferior. If “report-
1804 hazard” is “true”, the Inferior will reply with HAZARD if a mixed condition
1805 occurs, or if the Inferior cannot determine that a mixed condition has not
1806 occurred. If “report-hazard” is false, the Inferior will report only its own decision,
1807 regardless of whether that decision was correctly and consistently applied.
1808 Default is false.

1809
1810 **qualifiers** standardised or other qualifiers.

1811
1812 CONFIRM_ONE_PHASE can be issued by a Superior to an Inferior from whom
1813 PREPARED has been received (subject to the requirement that there is only one enrolled
1814 Inferior).

1815
1816 Types of FAULT possible (sent to “sender-Superior-address”)

1817
1818 *General*
1819 *InvalidInferior* – if “inferior-identifier” is unknown
1820 *WrongState* – if a PREPARE has already been sent to this Inferior

1821
1822 **HAZARD**

1823
1824 Sent when the Inferior has either discovered a “mixed” condition: that is unable to correctly
1825 and consistently cancel or confirm the operations in accord with the decision , or when the
1826 Inferior is unable to determine that a “mixed” condition has not occurred.

1827
1828 HAZARD is also used to reply to a CONFIRM_ONE_PHASE if the Inferior determines there
1829 is a mixed condition within its associated operations or is unable to determine that there is not
1830 a mixed condition.

1831

1832 Note - If the Inferior makes its own autonomous decision then it signals that
1833 decision with CONFIRMED or CANCELLED and waits to receive a
1834 confirmatory CONFIRM or CANCEL, or a CONTRADICTION if the
1835 autonomous decision by the Inferior was the opposite of that made by the
1836 Superior.

1837

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

1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873

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

[sender-address](#) the address from which the HAZARD is sent. This is an address of the Inferior.

superior-identifier The “superior-identifier” as on the ENROL message

inferior-identifier The “inferior-identifier” as on the earlier ENROL message

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

qualifiers standardised or other qualifiers.

Types of FAULT possible (sent to [“inferior-sender-address-as-inferior”](#))

General

InvalidSuperior – if “superior-identifier” is unknown

InvalidInferior – if no ENROL has been received for this [“inferior-address-as-inferior”](#) and [“inferior-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.

1874

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

1875

1876

target-address the address to which the CONTRADICTION message is sent. This will be the "[inferior-address](#) ~~as inferior~~" from the ENROL message.

1877

1878

1879

[sender-address](#) the address from which the CONTRADICTION is sent. This is an address of the Superior.

1880

1881

1882

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

1883

1884

1885

qualifiers standardised or other qualifiers.

1886

1887

Types of FAULT possible (sent to "[sender-Superior-address](#)")

1888

1889

General

1890

InvalidInferior – if "inferior-identifier" is unknown

1891

1892

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

1893

1894

SUPERIOR_STATE

1895

1896

Sent by a Superior as a query to an Inferior when

1897

1898

1. in the active state

1899

1900

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

1901

1902

1903

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

1904

1905

Parameter	Type
target-address	BTP address
sender-address	BTP address
inferior-identifier	Identifier
status	<i>see below</i>

Parameter	Type
response-requested	Boolean
qualifiers	List of qualifiers

1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918

target-address the address to which the SUPERIOR_STATE message is sent. This will be the “[inferior-address-as-inferior](#)” from the ENROL message.

[sender-address](#) the address from which the SUPERIOR_STATE is sent. This is an address of the Superior.

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

1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932

response-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. Default is “false”

qualifiers standardised or other qualifiers.

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

A status of unknown shall only be sent if it has been determined for certain that the Superior has no knowledge of the Inferior, or (equivalently) it can be determined that the relationship

1933 with the Inferior was cancelled. If there could be persistent information corresponding to the
1934 Superior, but it is not accessible from the entity receiving an INFERIOR_STATE/*y (or
1935 other) message targeted to the Superior or that entity cannot determine whether any such
1936 persistent information exists or not, the response shall be Inaccessible.

1937
1938 SUPERIOR_STATE/unknown is also used as a response to messages, other than
1939 INFERIOR_STATE/*y that are received when the Inferior is not known (and it is known
1940 there is no state information for it).

1941
1942 The form SUPERIOR_STATE/abcd refers to a SUPERIOR_STATE message status having a
1943 value equivalent to “abcd” (for active, prepared-received, unknown and inaccessible) and
1944 with “response-requested” = “false”. SUPERIOR_STATE/abcd/y refers to a similar message,
1945 but with “response-requested” = “true”. The form SUPERIOR_STATE/*y refers to a
1946 SUPERIOR_STATE message with “response-requested” = “true” and any value for status.

1947
1948

1949 INFERIOR_STATE

1950
1951 Sent by an Inferior as a query when in the active state to a Superior, when (due recovery from
1952 previous failure or other reason) there is uncertainty what state the Superior has reached.

1953
1954 Also sent by the Inferior to the Superior in response to a received SUPERIOR_STATE, in
1955 particular states.
1956

Parameter	Type
target-address	BTP address
sender-address	BTP address
superior-identifier	Identifier
inferior-identifier	Identifier
status	<i>see below</i>
response-requested	Boolean
qualifiers	List of qualifiers

1957
1958 **target-address** the address to which the INFERIOR_STATE is sent. This will
1959 be the “target-address” as used the original ENROL message.

1960
1961 [sender-address](#) the address from which the INFERIOR_STATE is sent. This is
1962 an address of the Inferior.

1963
1964 **superior-identifier** The “superior-identifier” as used on the ENROL message

1965
1966 **inferior-identifier** The “inferior-identifier” as on the ENROL message
1967

1968 **status** states the current state of the Inferior for the atomic business transaction,
1969 which corresponds to the last message sent to the Superior by (or in the case of
1970 ENROL for) the Inferior
1971

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

1972
1973 **response-requested** “true” if INFERIOR_STATE is sent as a query at the
1974 Superior’s initiative; “false” if INFERIOR_STATE is sent in reply to a received
1975 SUPERIOR_STATE or other message. Can only be “true” if “status” is “active”
1976 or “prepared-received”. Default is “false”
1977

1978 **qualifiers** standardised or other qualifiers.
1979

1980 The Superior, on receiving INFERIOR_STATE with “response-requested” = “true”, should
1981 reply in a timely manner by (depending on its state) repeating the previous message it sent or
1982 by sending SUPERIOR_STATE with the appropriate status value.
1983

1984 A status of “unknown” shall only be sent if it has been determined for certain that the Inferior
1985 has no knowledge of a relationship with the Superior. If there could be persistent information
1986 corresponding to the Superior, but it is not accessible from the entity receiving an
1987 SUPERIOR_STATE/*y (or other) message targetted on the Inferior or the entity cannot
1988 determine whether any such persistent information exists, the response shall be
1989 “inaccessible”.
1990

1991 INFERIOR_STATE/unknown is also used as a response to messages, other than
1992 SUPERIOR_STATE/*y that are received when the Inferior is not known (and it is known
1993 there is no state information for it).
1994

1995 A SUPERIOR_STATE/INFERIOR_STATE exchange that determines that one or both sides
1996 are in the active state does not require that the Inferior be cancelled (unlike some other two-
1997 phase commit protocols). The relationship between Superior and Inferior, and related
1998 application elements may be continued, with new application messages carrying the same
1999 CONTEXT. Similarly, if the Inferior is prepared but the Superior is active, there is no
2000 required impact on the progression of the relationship between them.
2001

2002 The form INFERIOR_STATE/abcd refers to a INFERIOR_STATE message status having a
2003 value equivalent to “abcd” (for active, unknown and inaccessible) and with “response-

2004 requested" = "false". INFERIOR_STATE/abcd/y refers to a similar message, but with
2005 "response-requested" = "true". The form INFERIOR_STATE/*/*y refers to a
2006 INFERIOR_STATE message with "response-requested" = "true" and any value for status.

2007

2008

2009 REDIRECT

2010

2011

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

2012

2013

2014

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

2015

2016

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

2017

2018

2019

2020

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

2021

2022

2023

2024

inferior-identifier The "inferior-identifier" as on the ENROL message

2025

2026

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

2027

2028

2029

2030

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

2031

2032

2033

qualifiers standardised or other qualifiers.

2034

2035

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

2036

2037

2038

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

2039

2040

2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051

Messages used in control relationships

BEGIN

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

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

2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073

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

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

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

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

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

2074
2075
2076
2077

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.

2078 The forms BEGIN/cohesion and BEGIN/atom refer to BEGIN with “transaction-type” having
2079 the corresponding value.

2080
2081 Types of FAULT possible (sent to “reply-address”)
2082

2083 **General**

2084 [Redirect](#) – *if the Factory now has a different address*

2085 **WrongState** - only issued if there is a related CONTEXT, and the
2086 Superior identified by the CONTEXT is in the wrong state to enrol new
2087 Inferiors
2088

2089 **BEGUN**

2090
2091 BEGUN is a reply to BEGIN. There is always a related CONTEXT, which is the CONTEXT
2092 for the new business transaction.
2093

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

2094
2095 **target-address** the address to which the BEGUN is sent. This will be the “reply-
2096 address” from the BEGIN.

2097
2098 [decider-address](#)~~-as-decider~~ for a top-most transaction (no CONTEXT related
2099 to the BEGIN), this is the address to which PREPARE_INFERIORS,
2100 CONFIRM_TRANSACTION, CANCEL_TRANSACTION,
2101 CANCEL_INFERIORS and REQUEST_INFERIOR_STATUSES messages are
2102 to be sent; if a CONTEXT was related to the BEGIN this parameter is absent
2103

2104 [inferior-address](#)~~-as-inferior~~ for a non-top-most transaction (a CONTEXT was
2105 related to the BEGIN), this is the “[inferior-address](#)~~-as-inferior~~” used in the
2106 enrolment with the Superior identified by the CONTEXT related to the BEGIN.
2107 The parameter is optional (implementor’s choice) if this is not a top-most
2108 transaction; it shall be absent if this is a top-most transaction.
2109

2110 **transaction-identifier** if this is a top-most transaction, this is an globally-
2111 unambiguous identifier for the new Decider (Composer or Coordinator). If this is
2112 not a top-most transaction, the transaction-identifier shall be the inferior-
2113 identifier used in the enrolment with the Superior identified by the CONTEXT
2114 related to the BEGIN.
2115

2116
2117

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

2118
2119
2120

qualifiers standardised or other qualifiers.

2121
2122
2123
2124
2125

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

2126
2127

No FAULT messages are issued on receiving BEGUN.

2128

PREPARE_INFERIORS

2129
2130

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

2131
2132
2133
2134
2135
2136

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

2137

2138
2139

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

2140

2141
2142

reply-address the address of the Terminator sending the PREPARE_INFERIORS message.

2143

2144
2145

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

2146

2147
2148

inferiors-list defines which of the Inferiors of this Decider preparation is requested for, using the “inferior-identifiers” as on the ENROL received by the Decider (in its role as Superior). If this parameter is absent, the PREPARE applies to all Inferiors.

2149

2150

2151

2152 **qualifiers** standardised or other qualifiers.

2153

2154

2155 For all Inferiors identified in the inferiors-list parameter (all Inferiors if the parameter is
2156 absent), from which none of PREPARED, CANCELLED or RESIGNED has been received,
2157 the Decider shall issue PREPARE. It will reply to the Terminator, using the “reply-address”
2158 on the PREPARE_INFERIORS message, sending an INFERIOR_STATUSES message
2159 giving the status of the Inferiors identified on the inferiors-list parameter (all of them if the
2160 parameter was absent).

2161

2162 If one or more of the “inferior-identifier”s in the "inferior-list" is unknown (does not
2163 correspond to an enrolled Inferior), a FAULT/Invalid-inferior shall be returned. The Decider
2164 shall not send PREPARE to any Inferior.

2165

2166 Types of FAULT possible (sent to Superior address)

2167

2168 **General**

2169 *InvalidDecider* – if Decider address is unknown

2170 *Redirect* – if the Decider- now has a different decider-address-as-decider

2171 *UnknownTransaction* – if the transaction-identifier is unknown

2172 *InvalidInferior* – if ~~an~~ one or more inferior-handles on the inferiors-list is

2173 unknown

2174 *WrongState* – if a CONFIRM_TRANSACTION or

2175 CANCEL_TRANSACTION has already been received by this

2176 Composer.

2177

2178 The form PREPARE_INFERIORS/all refers to a PREPARE_INFERIORS message where
2179 the “inferiors-list” parameter is absent. The form PREPARE_INFERIORS/specific refers to a
2180 PREPARE_INFERIORS message where the “inferiors-list” parameter is present.

2181

2182

2183 CONFIRM_TRANSACTION

2184

2185 Sent from a Terminator to a Decider to request confirmation of the business transaction. If the
2186 business transaction is a Cohesion, the confirm-set is specified by the “inferiors-list”
2187 parameter.

2188

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

2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224

2225
2226
2227
2228
2229

2230
2231

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

reply-address the address of the Terminator sending the CONFIRM_TRANSACTION message.

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

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

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

qualifiers standardised or other qualifiers.

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

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

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

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

2232 A confirm decision may be made only if PREPARED has been received from all Inferiors in
2233 the “confirm-set”. The making of the decision shall be persistent (and if it is not possible to
2234 persist the decision, it is not made). If there is only one remaining Inferior in the “confirm
2235 set” and PREPARE has not been sent to it, CONFIRM_ONE_PHASE may be sent to it.

2236
2237 All remaining Inferiors that are not in the confirm set shall be cancelled.

2238
2239 If a confirm decision is made and “report-hazard” was “false”, a
2240 [TRANSACTION_CONFIRMED_COMPLETE](#) message shall be sent to the “reply-address”.

2241
2242 If a cancel decision is made and “report-hazard” was “false”, a
2243 [TRANSACTION_CANCELLED_COMPLETE](#) message shall be sent to the “reply-address”.

2244
2245 If “report-hazard” was “true” and any HAZARD or contradictory message was received (i.e.
2246 CANCELLED from an Inferior in the confirm-set or CONFIRMED from an Inferior not in
2247 the confirm-set), an INFERIOR_STATUSES reporting the status for all Inferiors shall be sent
2248 to the “reply-address”.

2249
2250 [If one or more of the "inferior-identifier"s in the "inferior-list" is unknown \(does not
2251 correspond to an enrolled Inferior\), a FAULT/Invalid-inferior shall be returned. The Decider
2252 shall not make a confirm decision and shall not send CONFIRM to any Inferior.](#)

2253
2254 Types of FAULT possible (sent to “reply-address”)

2255
2256 **General**
2257 **InvalidDecider** – if Decider address is unknown
2258 [Redirect – if the Decider now has a different decider-address-as-decider](#)
2259 **UnknownTransaction** – if the transaction-identifier is unknown
2260 **InvalidInferior** – if ~~an~~ [one or more](#) inferior handles in the inferiors-list is
2261 unknown
2262 **WrongState** – if a CANCEL_TRANSACTION has already been
2263 received .

2264
2265 The form CONFIRM_TRANSACTION/all refers to a CONFIRM_TRANSACTION message
2266 where the “inferiors-list” parameter is absent. The form
2267 CONFIRM_TRANSACTION/specific refers to a CONFIRM_TRANSACTION message
2268 where the “inferiors-list” parameter is present.

2269 2270 TRANSACTION_CONFIRMED

2271
2272 A Decider sends TRANSACTION_CONFIRMED to a Terminator in reply to
2273 CONFIRM_TRANSACTION if all of the confirm-set confirms (and, for a Cohesion, all other
2274 Inferiors cancel) without reporting hazards, or if the Decider made a confirm decision and the
2275 CONFIRM_TRANSACTION had a “report-hazards” value of “false”.
2276

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

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

2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297

target-address the address to which the TRANSACTION_CONFIRMED is sent., this will be the “reply-address” from the CONFIRM_TRANSACTION message.

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

qualifiers standardised or other qualifiers.

Types of FAULT possible (sent to “[decider-address](#)~~-as-decider~~”)

General

InvalidTerminator – if Terminator address is unknown

UnknownTransaction – if the transaction-identifier is unknown

CANCEL_TRANSACTION

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

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

2298
2299
2300
2301
2302
2303
2304
2305
2306
2307

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

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

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

2308 **report-hazard** Defines whether the Terminator wishes to be informed of hazard
2309 events and contradictory decisions within the business transaction. If “report-
2310 hazard” is “true”, the receiver will wait until responses (CONFIRMED,
2311 CANCELLED or HAZARD) have been received from all of its inferiors,
2312 ensuring that any hazard events are reported. If “report-hazard” is “false”, the
2313 Decider will reply with TRANSACTION_CANCELLED immediately.

2314
2315 **qualifiers** standardised or other qualifiers.

2316
2317 The business transaction is cancelled – this is propagated to any remaining Inferiors by
2318 issuing CANCEL to them. No more Inferiors will be permitted to enrol.

2319
2320 Types of FAULT possible (sent to Superior address)

2321
2322 *General*
2323 *InvalidDecider* – if Decider address is unknown
2324 *Redirect – if the Decider now has a different decider-address-as-decider* |
2325 *UnknownTransaction* – if the transaction-identifier is unknown
2326 *WrongState* – if a CONFIRM_TRANSACTION has been received by
2327 this Composer.

2328

2329

2330 CANCEL_INFERIORS

2331

2332 Sent by a Terminator to a Decider, but only if is a Cohesion Composer, at any time before
2333 CONFIRM_TRANSACTION or CANCEL_TRANSACTION has been sent.

2334

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

2335

2336 **target-address** the address to which the CANCEL_TRANSACTION message is
2337 sent. This will be the decider-address from the BEGUN message.

2338

2339 **reply-address** the address of the Terminator sending the
2340 CANCEL_TRANSACTION message.

2341

2342 **transaction-identifier** identifies the Decider and will be the transaction-
2343 identifier from the BEGUN message.

2344

2345 **inferiors-list** defines which of the Inferiors of this Decider are to be cancelled,
2346 using the “inferior-identifiers” as on the ENROL received by the Decider (in its
2347 role as Superior).

2348
2349 **qualifiers** standardised or other qualifiers.

2350
2351
2352 Only the Inferiors identified in the inferiors-list are to be cancelled. Any other inferiors are
2353 unaffected by a CANCEL_INFERIORS. Further Inferiors may be enrolled.
2354

2355 Note – A CANCEL_INFERIORS [for](#) all of the currently enrolled Inferiors
2356 will leave the cohesion ‘empty’, but permitted to continue with new
2357 Inferiors, if any enrol.

2358
2359 [If one or more of the "inferior-identifier"s in the "inferior-list" is unknown \(does not](#)
2360 [correspond to an enrolled Inferior\), a FAULT/Invalid-inferior shall be returned. It is an](#)
2361 [implementation option whether CANCEL is sent to any of the Inferiors that are validly](#)
2362 [identified in the "inferiors-list".](#)

2363
2364 Types of FAULT possible (sent to Superior address)

2365
2366 **General**
2367 **InvalidDecider** – if Decider address is unknown
2368 **Redirect** – [if the Decider now has a different decider-address-as-decider](#)
2369 **UnknownTransaction** – if the transaction-identifier is unknown
2370 **InvalidInferior** – if ~~an~~ [one or more](#) inferior-handle on the inferiors-list is
2371 unknown
2372 **WrongState** – if a CONFIRM_TRANSACTION or
2373 CANCEL_TRANSACTION has been received by this Composer.

2374 2375 2376 2377 **TRANSACTION_CANCELLED**

2378
2379 A Decider sends TRANSACTION_CANCELLED to a Terminator in reply to
2380 CANCEL_TRANSACTION or in reply to CONFIRM_TRANSACTION if the Decider
2381 decided to cancel. In both cases, TRANSACTION_CANCELLED is used only if all Inferiors
2382 cancelled without reporting hazards or the CANCEL_TRANSACTION or
2383 CONFIRM_TRANSACTION had a “report-hazard” value of “false.”
2384

Parameter

target-address	BTP address
transaction-identifier	identifier

qualifiers List of qualifiers

2385

2386

target-address the address to which the TRANSACTION_CANCELLED is sent. This will be the “reply-address” from the CANCEL_TRANSACTION or CONFIRM_TRANSACTION message.

2387

2388

2389

2390

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

2391

2392

qualifiers standardised or other qualifiers.

2393

2394

Types of FAULT possible (sent to “[decider-address-as-decider](#)”)

2395

2396

General

2397

InvalidTerminator – if Terminator address is unknown

2398

UnknownTransaction – if the transaction-identifier is unknown

2399

2400

2401

REQUEST_INFERIOR_STATUSES

2402

2403

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

2404

2405

2406

2407

2408

2409

2410

Parameter

Type

target-address

BTP address

reply-address

BTP address

target-identifier

Identifier

inferiors-list

List of Identifiers

qualifiers

List of qualifiers

2411

2412

target-address the address to which the REQUEST_STATUS message is sent. When used to a Decider, this will be the “[decider-address-as-decider](#)” from the BEGUN message. Otherwise it may be an “[superior-address-as-superior](#)” from a CONTEXT or “[inferior-address-as-inferior](#)” from an ENROL message.

2413

2414

2415

2416

2417

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

2418

2419

2420 **target-identifier** identifies the transaction (or transaction tree node). When the
 2421 message is used to a Decider, this will be the transaction-identifier from the
 2422 BEGUN message. Otherwise it will be the superior-identifier from a CONTEXT
 2423 or an inferior-identifier from an ENROL message.
 2424
 2425 **inferiors-list** defines which inferiors enrolled with the target are to be included
 2426 in the INFERIOR_STATUSES, using the “inferior-identifiers” as on the ENROL
 2427 received by the Decider (in its role as Superior). If the list is absent, the status of
 2428 all enrolled Inferiors will be reported.
 2429
 2430 **qualifiers** standardised or other qualifiers.

2431
 2432 Types of FAULT possible (sent to reply-address)
 2433

2434 **General**

2435 **Redirect** – if the intended target now has a different address

2436 **StatusRefused** – if the receiver is not prepared to report its status to the
 2437 sender of this message. This “fault-type” shall not be issued when a Decider
 2438 receives REQUES_STATUSES from the Terminator.

2439 **UnknownTransaction** – if the transaction-identifier is unknown
 2440

2441
 2442 The form REQUEST_INFERIOR_STATUSES/all refers to a REQUEST_STATUS with the
 2443 inferiors-list absent. The form REQUEST_INFERIOR_STATUS/specific refers to a
 2444 REQUEST_INFERIOR_STATUS with the inferiors-list present.
 2445

2446 **INFERIOR_STATUSES**

2447
 2448 Sent by a Decider to report the status of all or some of its inferiors in response to a
 2449 REQUEST_INFERIOR_STATUSES, PREPARE_INFERIORS, CANCEL_INFERIORS,
 2450 CANCEL_TRANSACTION with “report-hazard” value of “true” and
 2451 CONFIRM_TRANSACTION with “report-hazard”value of “true”. It is also used by any
 2452 actor in response to a received REQUEST_INFERIOR_STATUSES to report the status of
 2453 inferiors, if there are any.
 2454

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

2455
 2456 **target-address** the address to which the INFERIOR_STATUSES is sent. This
 2457 will be the “reply-address” on the received message
 2458

2459
2460
2461
2462
2463
2464

responders-identifier the target-identifier used on the REQUEST_INFERIOR_STATUSES.

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

Field	Type
Inferior-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).

2465
2466
2467
2468

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

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

status value	Meaning
<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)

2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506

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 “[decider-address](#)” ~~as decider~~)

General

InvalidTerminator – if Terminator address is unknown

UnknownTransaction – if the transaction-identifier is unknown

Groups – combinations of related messages

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.

CONTEXT & application message

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.

2507 **target-address:** the “target-address” is that of the application message. It is not required
2508 that the application address be a BTP address (in particular, there is no BTP-defined
2509 “additional information” field – the application protocol (and its binding) may or may not
2510 have a similar construct).

2511
2512 There may be multiple application messages related to a single CONTEXT message. All
2513 the application messages so related are deemed to be part of the business transaction
2514 identified by the CONTEXT. This specification does not imply any further relatedness
2515 among the application messages themselves (though the application might).

2516
2517 The actor that sends the group shall retain knowledge of the Superior address in the
2518 CONTEXT. If the CONTEXT is a CONTEXT/atom, the actor shall also keep track of
2519 transmitted CONTEXTs for which no CONTEXT_REPLY has been received.

2520
2521 If the CONTEXT is a CONTEXT/atom, the actor receiving the CONTEXT shall ensure
2522 that a CONTEXT_REPLY message is sent back to the “reply-address” of the CONTEXT
2523 with the appropriate completion status.
2524

2525 Note – The representation of the relation between CONTEXT and one or
2526 more application messages depends on the binding to the carrier protocol. It
2527 is not necessary that the CONTEXT and application messages be closely
2528 associated “on the wire” (or even sent on the same connection) – some kind
2529 of referencing mechanism may be used.

2530 2531 **CONTEXT_REPLY & ENROL**

2532
2533 **Meaning:** the enrolment of the Inferior identified in the ENROL is to be performed with
2534 the Superior identified in the CONTEXT message this CONTEXT_REPLY is replying
2535 to. If the “completion-status” of CONTEXT_REPLY is “related”, failure of this
2536 enrolment shall prevent the confirmation of the business transaction.

2537
2538 **target-address:** the “target-address” is that of the CONTEXT_REPLY. This will be the
2539 “reply-address” of the CONTEXT message (in many cases, including request/reply
2540 application exchanges, this address will usually be implicit).

2541
2542 The “target-address” of the ENROL message is omitted.

2543
2544 The actor receiving the related group will use the retained Superior address from the
2545 CONTEXT sent earlier to forward the ENROL. When doing so, it changes the ENROL to
2546 ask for a response (if it was an ENROL/no-rsp-req) and supplies its own address as the
2547 “reply-address”, remembering the original “reply-address” if there was one.

2548
2549 If ENROLLED is received and the original received ENROL was ENROL/rsp-req, the
2550 ENROLLED is forwarded back to the original “reply-address”.
2551

2552 If this attempt fails (i.e. ENROLLED is not received), and the “completion-status” of the
2553 CONTEXT_REPLY was “related”, the actor is required to ensure that the Superior does
2554 not proceed to confirmation. How this is achieved is an implementation option, but must
2555 take account of the possibility that direct communication with the Superior may fail. (One
2556 method is to prevent CONFIRM_TRANSACTION being sent to the Superior (in its role
2557 as Decider); another is to enrol as another Inferior before sending the original CONTEXT
2558 out with an application message). If the Superior is a sub-coordinator or sub-composer,
2559 an enrolment failure must ensure the sub-coordinator does not send PREPARED to its
2560 own Superior.

2561
2562 If the actor receiving the related group is also the Superior (i.e. it has the same binding
2563 address), the explicit forwarding of the ENROL is not required, but the resultant effect –
2564 that if enrolment fails the Superior does not confirm or issue PREPARED – shall be the
2565 same.

2566
2567 A CONTEXT_REPLY & ENROL group may contain multiple ENROL messages, for
2568 several Inferiors. Each ENROL shall be forwarded and an ENROLLED reply received
2569 before the Superior is allowed to confirm if the “completion-status” in the
2570 CONTEXT_REPLY was “related”.

2571
2572 When the group is constructed, if the CONTEXT had “superior-type” value of “atom”,
2573 the “completion-status” of the CONTEXT_REPLY shall be “related”. If the “superior-
2574 type” was “cohesive”, the “completion-status” shall be “completed” or “related” (as
2575 required by the application). If the value is “completed”, the actor receiving the group
2576 shall forward the ENROLs, but is not required to (though it may) prevent confirmation.
2577

2578 CONTEXT_REPLY (& ENROL) & PREPARED / & CANCELLED

2579
2580 This combination is characterised by a related CONTEXT_REPLY and either or both of
2581 PREPARED and CANCELLED, with or without ENROL.

2582
2583 **Meaning:** If ENROL is present, the meaning and required processing is the same as for
2584 CONTEXT_REPLY & ENROL. The PREPARED or CANCELLED message(s) are
2585 forwarded to the Superior identified in the CONTEXT message this CONTEXT_REPLY
2586 is replying to.
2587

2588 Note – the combination of CONTEXT_REPLY & ENROL & CANCELLED
2589 may be used to force cancellation of an atom

2590
2591 **target-address:** the “target-address” is that of the CONTEXT_REPLY. This will be the
2592 “reply-address” of the CONTEXT message (in many cases, including request/reply
2593 application exchanges, this address will usually be implicit).
2594

2595 The “target-address” of the PREPARED and CANCELLED message is omitted – they
2596 will be sent to the Superior identified in the earlier CONTEXT message.

2597
2598 The actor receiving the group forwards the PREPARED or CANCELLED message to the
2599 Superior in as for an ENROL, using the retained Superior address from the CONTEXT
2600 sent earlier, except there is no reply required from the Superior.
2601
2602 If (as is usual) an ENROL and PREPARED or CANCELLED message are for the same
2603 Inferior, the ENROL shall be sent first, but the actor need not wait for the ENROLLED to
2604 come back before sending the PREPARED or CANCELLED (so an
2605 ENROL+PREPARED bundle from this actor to the Superior could be used).
2606
2607 The group can contain multiple ENROL, PREPARED and CANCELLED messages.
2608 Each PREPARED and CANCELLED message will be for a different Inferior.. There is
2609 no constraint on the order of their forwarding, except that ENROL and PREPARED or
2610 CANCELLED for the same Inferior shall be delivered to the Superior in the order
2611 ENROL first, followed by the other message for that Inferior.
2612
2613
2614

2615 **CONTEXT_REPLY & ENROL & application message (& PREPARED)**

2616
2617 This combination is characterised by a related CONTEXT_REPLY, ENROL and an
2618 application message. PREPARED may or may not be present in the related group.
2619

2620 **Meaning:** the relation between the BTP messages is as for the preceding groups, The
2621 transmission of the application message (and application effects implied by its
2622 transmission) has been associated with the Inferior identified by the ENROL and will be
2623 subject to the outcome delivered to that Inferior.
2624

2625 **target-address:** the “target-address” of the group is the “target-address” of the
2626 CONTEXT_REPLY which shall also be the “target-address” of the application message.
2627 The ENROL and PREPARED messages do not contain their “target-address” parameters.
2628

2629 The processing of ENROL and PREPARED messages is the same as for the previous
2630 groups.
2631

2632 This group can be used when participation in business transaction (normally a cohesion),
2633 is initiated by the service (Inferior) side, which fetches or acquires the CONTEXT, with
2634 some associated application semantic, performs some work for the transaction and sends
2635 an application message with a related ENROL. The CONTEXT_REPLY allows the
2636 addressing of the application (and the CONTEXT_REPLY) to be distinct from that of the
2637 Superior.
2638

2639 The actor receiving the group may associate the “inferior-identifier” received on the
2640 ENROL with the application message in a manner that is visible to the application
2641 receiving the message (e.g. for subsequent use in Terminator:Decider exchanges).
2642

2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686

BEGUN & CONTEXT

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

target-address: the “target-address” is that of the BEGUN message – this will be the “reply-address” of the earlier BEGIN message.

BEGIN & CONTEXT

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

target-address: the “target-address” is that of the BEGIN – this will be the address of the Factory.

Standard qualifiers

The following qualifiers are expected to be of general use to many applications and environments. The URI “urn:oasis:names:tc:BTP:1.0:qualifiers” is used in the Qualifier group value for the qualifiers defined here.

Transaction timelimit

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

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

The qualifier is propagated on a CONTEXT message.

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

The “Content” shall contain the following field:

Content field	Type
Timelimit	Integer

2687 **Timelimit** indicates the maximum (further) duration, expressed as whole seconds from the
2688 time of transmission of the containing CONTEXT, of the active phase of the business
2689 transaction.

2690
2691 **Inferior timeout**

2692
2693 This qualifier allows an Inferior to limit the duration of its “promise”, when sending
2694 PREPARED, that it will maintain the ability to confirm or cancel the effects of all associated
2695 operations. Without this qualifier, an Inferior is expected to retain the ability to confirm or
2696 cancel indefinitely. If the timeout does expire, the Inferior is released from its promise and
2697 can apply the decision indicated in the qualifier.

2698
2699 It should be noted that BTP recognises the possibility that an Inferior may be forced to apply
2700 a confirm or cancel decision before the CONFIRM or CANCEL is received and before this
2701 timeout expires (or if this qualifier is not used). Such a decision is termed a heuristic decision,
2702 and (as with other transaction mechanisms), is considered to be an exceptional event. As with
2703 heuristic decisions, the taking of an autonomous decision by a Inferior **subsequent** to the
2704 expiry of this timeout, is liable to cause contradictory decisions across the business
2705 transaction. BTP ensures that at least the occurrence of such a contradiction will be
2706 (eventually) reported to the Superior of the business transaction. BTP treats “true” heuristic
2707 decisions and autonomous decisions after timeout the same way – in fact, the expiry in this
2708 timeout does not cause a qualitative (state table) change in what can happen, but rather a step
2709 change in the probability that it will.

2710
2711 The expiry of the timeout does not strictly require that the Inferior immediately invokes the
2712 intended decision, only that is at liberty to do so. An implementation may choose to only
2713 apply the decision if there is contention for the underlying resource, for example.
2714 Nevertheless, Superiors are recommended to avoid relying on this and ensure decisions for
2715 the business transaction are made before these timeouts expire (and allow a margin of error
2716 for network latency etc.).

2717
2718 The qualifier may be present on a PREPARED message. If the PREPARED message has the
2719 “default-is cancel” parameter “true”, then the “IntendedDecision” field of this qualifier shall
2720 have the value “cancel”.

2721
2722 The “Qualifier name” shall be “inferior-timeout” .

2723
2724 The “Content” shall contain the following fields:

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

2726
2727 **Timeout** indicates how long, expressed as whole seconds from the time of transmission of the
2728 carrying message, the Inferior intends to maintain its ability to either confirm or cancel the
2729 effects of the associated operations, as ordered by the receiving Superior.

2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751

2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773

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

Minimum inferior timeout

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

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

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

The “Content” shall contain the following field:

Content field	Type
MinimumTimeout	Integer

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

Inferior name

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

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

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

2774 The qualifier may be present on BEGIN, ENROL and in the “qualifiers” field of a Status-item
2775 in INFERIOR_STATUSES. It is present on BEGIN only if there is a related CONTEXT; if
2776 present, the same qualifier value **should** be included in the consequent ENROL. If
2777 INFERIOR_STATUSES includes a Status-item for an Inferior whose ENROL had an
2778 inferior-name qualifier, the same qualifier value **should** be included in the Status-item.
2779

2780 The “Qualifier -name” shall be “inferior-name”

2781

2782 The “Content” shall contain the following fields:

2783

Content field	Type
inferior-name	String

2784

2785 **Inferior name** the name assigned to the enrolling Inferior.

2786

2787

State Tables

2788

Explanation of the state tables

2789

2790

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).

2791

2792

2793

2794

2795

2796

2797

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.

2798

2799

2800

2801

2802

2803

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.

2804

2805

2806

Status queries

2807

2808

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.

2809

2810

2811

2812

2813

2814

2815

2816

2817

2818

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 “response-requested” equal to “false” are only sent when the other message with “response-requested” equal to “true” has been received and no other message has been sent.

2819

2820

2821

2822

2823

2824

2825

2826

2827

Decision events

2828

2829

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

2830

2831

2832

2833 business transaction and on features of the implementation (e.g. making of a persistent record
2834 of the decision means that the information will survive at least some failures that otherwise
2835 lose state information, but the level of survival depends on the purpose of the
2836 implementation). [Table 2](#)~~Table 2~~~~Table 2~~~~Table 2~~ and [Table 3](#)~~Table 3~~~~Table 3~~~~Table 3~~ define the
2837 decision events.
2838

2839 In some cases, an implementation may not need to make an active change to have a persistent
2840 record of a decision, provided that the implementation will restore itself to the appropriate
2841 state on recovery. For example, an (inferior) implementation that “decided to be prepared”,
2842 and recorded a timeout (to cancel) in the persistent information for that decision (signalled via
2843 the appropriate qualifier on PREPARED), could treat the presence of an expired record as a
2844 record of “decide to cancel autonomously”, provided it always updated such a record as part
2845 of the “apply ordered confirmation” decision event.
2846

2847 The Superior event “decide to prepare” is considered semi-persistent. Since the sending of
2848 PREPARE indicates that the application exchange (to associate operations with the Inferior)
2849 is complete, it is not meaningful for the Superior:Inferior relationship to revert to an earlier
2850 state corresponding to an incomplete application exchange. However, implementations are
2851 not required to make the sending of PREPARE persistent in terms of recovery – a Superior
2852 that experiences failure after sending PREPARE may, on recovery, have no information
2853 about the transaction, in which case it is considered to be in the completed state (Z), which
2854 will imply the cancellation of the Inferior and its associated operations.
2855

2856 Where a Superior is itself an Inferior (to another Superior entity), in a hierarchic tree, its
2857 “decide to confirm” and “decide to cancel” decisions will in fact be the receipt of a
2858 CONFIRM or CANCEL instruction from its own Superior, without necessary change of local
2859 persistent information (which would combine both superior and inferior information, pointing
2860 both up and down the tree).
2861

2862 **Disruptions – failure events**

2863 Failure events are modelled as “disruption”. A failure and the subsequent recovery will (or
2864 may) cause a change of state. The disruption events in the state tables model different extents
2865 of loss of state information. An implementation is not required to exhibit all the possible
2866 disruption events, but it is not allowed to exhibit state transitions that do not correspond to a
2867 possible disruption.
2868

2869 In addition to the disruption events in the tables, there is an implicit “disruption 0” event,
2870 which involves possible interruption of service and loss of messages in transit, but no change
2871 of state (either because no state information was lost, or because recovery from persistent
2872 information restores the implementation to the same state). The “disruption 0” event would
2873 typically be an appropriate abstraction for a communication failure.
2874

2875 **Invalid cells and assumptions of the communication mechanism**

2876
2877
2878

2879 The empty cells in state table represent events that cannot happen. For events corresponding
2880 to sending a message or any of the decision events, this prohibition is absolute – e.g. a
2881 conformant implementation in the Superior active state “B1” will not send CONFIRM. For
2882 events corresponding to receiving a message, the interpretation depends on the properties of
2883 the underlying communications mechanism.
2884

2885 For all communication mechanisms, it is assumed that

- 2886 a) the two directions of the Superior:Inferior communication are not synchronised –
2887 that is messages travelling in opposite directions can cross each other to any
2888 degree; any number of messages may be in transit in either direction; and
- 2889 b) messages may be lost arbitrarily
2890

2891 If the communication mechanisms guarantee ordered delivery (i.e. that messages, if delivered
2892 at all, are delivered to the receiver in the order they were sent) , then receipt of a message in a
2893 state where the corresponding cell is empty indicates that the far-side has sent a message out
2894 of order – a FAULT message with the “fault-type” “WrongState” can be returned.
2895

2896 If the communication mechanisms cannot guarantee ordered delivery, then messages received
2897 where the corresponding cell is empty should be ignored. Assuming the far-side is
2898 conformant, these messages can assumed to be “stale” and have been overtaken by messages
2899 sent later but already delivered. (If the far-side is non-conformant, there is a problem
2900 anyway).
2901

2902 **Meaning of state table events**

2903
2904 The tables in this section define the events (rows) in the state tables. [Table 1](#)~~Table 1~~
2905 ~~Table 1~~ defines the events corresponding to sending or receiving BTP messages and the
2906 disruption events. [Table 2](#)~~Table 2~~
2907 ~~Table 2~~ describes the decision events for an Inferior, [Table 3](#)~~Table 3~~
2908 ~~Table 3~~ those for a Superior.

2909 The decision events for a Superior, defined in [Table 3](#)~~Table 3~~
2910 ~~Table 3~~ cannot be specified without reference to other Inferiors to which it is Superior and to its relation with
2911 the application or other entity that (acting ultimately on behalf of the application) drives it.
2912

2913 The term “remaining Inferiors” refers to any actors to which this endpoint is Superior and
2914 which are to be treated as an atomic decision unit with (and thus including) the Inferior on
2915 this relationship. If the CONTEXT for this Superior:Inferior relationship had a “superior-
2916 type” of “atom”, this will be all Inferiors established with same Superior address and
2917 “superior-identifier” except those from which RESIGN has been received. If the CONTEXT
2918 had “superior-type” of “cohesion”, the “remaining Inferiors” excludes any that it has been
2919 determined will be cancelled, as well as any that have resigned – in other words it includes
2920 only those for which a confirm decision is still possible or has been made. The determination
2921 of exactly which Inferiors are “remaining Inferiors” in a cohesion is determined, in some
2922 way, by the application. The term “Other remaining Inferiors” excludes this Inferior on this
2923 relationship. A Superior with a single Inferior will have no “other remaining Inferiors”.
2924

2925 In order to ensure that the confirmation decision **is** delivered to all remaining Inferiors,
 2926 despite failures, the Superior must persistently record which these Inferiors are (i.e. their
 2927 addresses and identifiers). It must also either record that the decision is confirm, or ensure
 2928 that the confirm decision (if there is one) is persistently recorded somewhere else, and that it
 2929 will be told about it. This latter would apply if the Superior were also BTP Inferior to another
 2930 entity which persisted a confirm decision (or recursively deferred it still higher). However,
 2931 since there is no requirement that the Superior be also a BTP Inferior to any other entity, the
 2932 behaviour of asking another entity to make (and persist) the confirm decision is termed
 2933 "offering confirmation" - the Superior offers the possible confirmation of itself, and its
 2934 remaining Inferiors to some other entity. If that entity (or something higher up) then does
 2935 make and persist a confirm decision, the Superior is "instructed to confirm" (which is
 2936 equivalent BTP CONFIRM).

2937
 2938 The application, or an entity acting indirectly on behalf of the application, may request a
 2939 Superior to prepare an Inferior (or all Inferiors). This typically implies that there will be no
 2940 more operations associated with the Inferior. Following a request to prepare all remaining
 2941 Inferiors, the Superior may offer confirmation to the entity that requested the prepare. (If the
 2942 Superior is also a BTP Inferior, its superior can be considered an entity acting on behalf of the
 2943 application.)

2944
 2945 The application, or an entity acting indirectly on behalf of the application, may also request
 2946 confirmation. This means the Superior is to attempt to make and persist a confirm decision
 2947 itself, rather than offer confirmation.

2948
 2949

2950

Table 1 : send, receive and disruption events

Event name	Meaning
send/receive ENROL/rsp-req	send/receive ENROL with response-requested = true
send/receive ENROL/no-rsp-req	send/receive ENROL with response-requested = false
send/receive RESIGN/rsp-req	send/receive RESIGN with response-requested = true
send/receive RESIGN/no-rsp-req	send/receive RESIGN with response-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 response-requested = true

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

2951

2952

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)

2953

2954

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)

2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983

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

2987

2988

2989

2990

2991

2992

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										

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								

2997

2998

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		

2999

3000

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

3001

3002

3002

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 s3 h2 h2 l1 l2		j1 j1 s4 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

3003

3004

3004

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		

3005

3006

3006

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	

3007

3008

3008

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

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

3009

3010

3010 **Failure Recovery**

3011 **Types of failure**

3012

3013 BTP is designed to ensure the delivery of a consistent decision for a business transaction to
3014 the parties involved, even in the event of failure. Failures can be classified as:

3015

3016

3017

3018

3019

3020

3021

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.

3022

3023

3024

3025

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.

3026

3027

3028

3029

3030

3031

3032

3033

3034

3035

3036

3037

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.

3038

3039

3040

3041

3042

3043

3044

3045

3046

3047

3048

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.

3049

3050

3051

3052

3053

3054

3055

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.

3056 After recovery from node failure, the implementation behaves much as if a communication
3057 failure had occurred.

3058

3059 **Persistent information**

3060

3061 BTP requires that some decision events are persisted – that information recording an
3062 Inferior’s decision to be prepared, a Superior’s decision to confirm and an Inferior’s
3063 autonomous decision survive failure. Making the first two decisions persistent ensures that a
3064 consistent decision can be reached for the business transaction and that it is delivered to all
3065 involved nodes. Requiring an Inferior’s autonomous decision to be persistent allows BTP to
3066 ensure that, if this decision is contradictory (i.e. opposite to the decision at the Superior), the
3067 contradiction will be reported to the Superior, despite failures.

3068

3069 BTP also permits, but does not require, recovery of the Superior:Inferior relationship in the
3070 active state (unlike many transaction protocols, where a communication or endpoint failure in
3071 active state would invariably cause rollback of the transaction). Recovery in the active state
3072 may require that the application exchange is resynchronised as well – BTP does not directly
3073 support this, but does allow continuation of the business transaction as such. In the state
3074 tables, from some states, there are several levels of disruption, distinguished by which state
3075 the implementation transits to – this represents the survival of different extents of state
3076 information over failure and recovery. The different levels of disruption describe legitimate
3077 states for the endpoint to be in after it has recovered – **they do not require that all
3078 implementations are able to exhibit the appropriate partial loss of state information.**

3079

3080 The absence of a destination state for the disruption events means that such a transition is not
3081 legitimate – thus, for example, an Inferior that has decided to be prepared will always recover
3082 to the same state, by virtue of the information persisted in the “decide to be prepared” event.

3082

3083 Apart from the (optional) recovery in active state, BTP follows the well-known presume-
3084 abort model – it is only required that information be persisted when decisions are made (and
3085 not, e.g. on enrolment). This means that on recovery, one side may have persistent
3086 information but the other does not. This occurs when an Inferior has decided to be prepared
3087 but the Superior never confirmed (so the decision is “presumed” to be cancel), or because the
3088 Superior did confirm, and the Inferior applied the confirm, removed its persistent information
3089 but the acknowledgement (CONFIRMED) was never received by the Superior (or, at least, it
3090 still had the persistent information when the failure occurred).

3091

3092 Information to be persisted for an Inferior’s “decision to be prepared” must be sufficient to
3093 re-establish communication with the Superior, to apply a confirm decision and to apply a
3094 cancel decision. It will thus need to include

3095

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

3096

Superior address (as on CONTEXT)

3097

“superior-identifier” (as on CONTEXT)

3098

default-is-cancel value (as on PREPARED)

3099

3100 The information needed to apply confirm/cancel decisions will depend on the application and
3101 the associated operations. It may also normally be necessary to persist any qualifiers that

3102 were sent with the PREPARED message or application messages sent with the PREPARED,
3103 since the PREPARED message will be repeated if a failure occurs.

3104

3105 A Superior must record corresponding information to allow it to re-establish communication
3106 with the Inferior:

3107 Inferior address (as on ENROL)

3108 “inferior-identifier” (as on ENROL)

3109

3110 A Superior that is the Decider for the business transaction need only persist this information
3111 if it makes a decision to confirm (and this Inferior is in the confirm set, for a Cohesion). A
3112 Superior that is also an Inferior to some other entity (i.e. it is an intermediate in a tree, as
3113 atom in a cohesion, sub-coordinator or sub-composer) must persist this information as
3114 Superior (to this Inferior) as part of the persistent information of its decision to be prepared
3115 (as an Inferior). For such an entity, the “decision to confirm” as Superior is made when (and
3116 if) CONFIRM is received from its Superior or it makes an autonomous decision to confirm. If
3117 CONFIRM is received, the persistent information may be changed to show the confirm
3118 decision, but alternatively, the receipt of the CONFIRM can be treated as the decision itself.
3119 If the persistent information is left unchanged and there is a node failure, on recovery the
3120 entity (as an Inferior) will be in a prepared state, and will rediscover the confirm decision
3121 (using the recovery exchanges to its Superior) before propagating it to its Inferior(s).

3122

3123 After failure, an implementation may not be able to restore an endpoint to the appropriate
3124 state immediately – in particular, the necessary persistent information may be inaccessible,
3125 although the implementation can respond to received BTP messages. In such a case, a
3126 Superior may reply to any BTP message except INFERIOR_STATE/* (i.e. with a “response-
3127 requested” value “false”) with SUPERIOR_STATE/inaccessible and an Inferior to any BTP
3128 message except SUPERIOR_STATE/* with “INFERIOR_STATE/inaccessible. Receipt of
3129 the *_STATE/inaccessible messages has no effect on the endpoint state.

3130

3131 **Redirection**

3132

3133 As described above, BTP uses the presume-abort model for recovery. A corollary of this is
3134 that there are cases where one side will attempt to re-establish communication when there is
3135 no persistent information for the relationship at the far-end. In such cases, it is important the
3136 side that is attempting recovery can distinguish between unsuccessful attempts to connect to
3137 the holder of the persistent information and when the information no longer exists. If the peer
3138 information does not exist, this side can draw conclusions and complete appropriately; if they
3139 merely fail to get through they are stuck in attempting recovery.

3140

3141 Two mechanisms are provided to make it possible that even when one side of a
3142 Superior:Inferior relationship has completed, that a message can eventually get through to
3143 something that can definitively report the status, distinguishing this case from a temporary
3144 inability to access the state of a continuing transaction element. The mechanisms are:

- 3145 o Address fields which provide a “callback address” can be a set of addresses,
3146 which are alternatives one of which is chosen as the “target-address” for the
3147 future message. If the sender of that message finds the address does not work,
3148 it can try a different alternative.

3149 o The REDIRECT message can be used to inform the peer that an address
3150 previously given is no longer valid and to supply a replacement address (or
3151 set of addresses). REDIRECT can be issued either as a response to receipt of
3152 a message or spontaneously.
3153

3154 The two mechanisms can be used in combination, with one or more of the original set of
3155 addresses just being a redirector, which does not itself ever have direct access to the state
3156 information for the transaction, but will respond to any message with an appropriate
3157 REDIRECT.

3158 An alternative implementation approach is to have a single addressable entity that uses the
3159 same address for all transactions, distinguishing them by identifier, and which always
3160 recovers to use the same address. Such an implementation would not need to supply
3161 “backup” addresses (and would only use REDIRECT if it was being permanently migrated).
3162

3163 **Terminator:Decider failures**

3164 BTP does not provide facilities or impose requirements on the recovery of
3165 Terminator:Decider relationships, other than allowing messages to be repeated. A Terminator
3166 may survive failures (by retaining knowledge of the Decider’s address and identifier), but this
3167 is an implementation option. Although a Decider (if it decides to confirm) will persist
3168 information about the confirm decision, it is not required, after failure, to remain accessible
3169 using the inferior address it offered to the Terminator. Any such recovery is an
3170 implementation option.
3171

3172 A Decider’s address (as returned on BEGUN) may be a set of addresses, allowing a failed
3173 Decider to be recovered at a different address.
3174

3175 A Decider has no way of initiating a call to a Terminator to ensure that it is still active, and
3176 thus no way of detecting that a Terminator has failed. To avoid a Decider waiting for ever for
3177 a CONFIRM_TRANSACTION that will never arrive, the standard qualifier “Transaction
3178 timelimit” can be used (by the Initiator) to inform the Decider when it can assume the
3179 Terminator will not issue CONFIRM_TRANSACTION and so it (the Decider) should initiate
3180 cancellation.
3181

3182 **XML representation of Message Set**

3183 This section describes the syntax for BTP messages in XML. These XML messages represent
3184 a midpoint between the abstract messages and what actually gets sent on the wire.
3185

3186 All BTP related URIs have been created using Oasis URI conventions as specified in [RFC](#)
3187 [3121](#)
3188

3189 The XML Namespace for the BTP messages is urn:oasis:names:tc:BTP:[1.0:corexml](#) |

3190 In addition to an XML schema, this specification uses an informal syntax to describe the
3191 structure of the BTP messages. The syntax appears as an XML instance, but the values
3192

3196 contain data types instead of values. The following symbols are appended to some of the
3197 XML constructs: ? (zero or one), * (zero or more), + (one or more.) The absence of one of
3198 these symbols corresponds to "one and only one."
3199

3200 Addresses

3201
3202 As described in the "Abstract Message and Associated Contracts – Addresses" section, a BTP
3203 address comprises three parts, and for a "target-address" only the "additional information"
3204 field is inside the BTP messages. For all BTP messages whose abstract form includes a
3205 "target-address" parameter, the corresponding XML representation includes a "target-
3206 additional-information" element. This element may be omitted if it would be empty.
3207

3208 For other addresses, all three fields are represent, as in:
3209

```
3210 <ctp:some-address>  
3211   <ctp:binding-name>...carrier binding URI...</ctp:binding-name>  
3212   <ctp:binding-address>...carrier specific  
3213   address...</ctp:binding-address>  
3214   <ctp:additional-information>...optional additional addressing  
3215   information...</ctp:additional-information> ?  
3216 </ctp:some-address>
```

3217
3218 A "published" address can be a set of <some-address>, which are alternatives which can be
3219 chosen by the peer (sender.) Multiple addresses are used in two cases: different bindings to
3220 same endpoint, or backup endpoints. In the former, the receiver of the message has the choice
3221 of which address to use (depending on which binding is preferable.) In the case where
3222 multiple addresses are used for redundancy, a priority attribute can be specified to help the
3223 receiver choose among the addresses- the address with the highest priority should be used,
3224 other things being equal. The priority is used as a hint and does not enforce any behaviour in
3225 the receiver of the message. Default priority is a value of 1.
3226
3227

3228 Qualifiers

3229 The "Qualifier name" is used as the element name, within the namespace of the "Qualifier
3230 group".
3231

3232 Examples:

```
3233 <ctp:inferior-timeout  
3234   xmlns:ctp="urn:oasis:names:tc:BTP:1.0:qualifiers"  
3235   xmlns:ctp="urn:oasis:names:tc:BTP:1.0:corexml"  
3236   ctp:must-be-understood="false"  
3237   ctp:to-be-propagated="false">1800</ctp:inferior-timeout>  
3238  
3239 <auth:username  
3240   xmlns:auth="http://www.example.com/ns/auth"  
3241   xmlns:ctp="urn:oasis:names:tc:BTP:1.0:corexml"  
3242   ctp:must-be-understood="true"  
3243   ctp:to-be-propagated="true">jtauber</auth:username>  
3244
```

3245 Attributes must-be-understood **has default value “true”** and to-be-propagated has default
3246 value “false”.

3247

3248 Identifiers

3249

3250 Identifiers shall be URIs "

3251

3252 Note – Identifiers need to be globally unambiguous. Apart from their
3253 generation, the only operation the BTP implementations have to perform on
3254 identifiers is to match them.

3255

3256 Message References

3257 Each BTP message has an optional id attribute to give it a unique identifier. An application
3258 can make use of those identifiers, but no processing is enforced.

3259

3260 Messages

3261

3262 CONTEXT

3263

```
3264 <btpt:context id?>  
3265 <btpt:superior-address> +  
3266 ...address...  
3267 </btpt:superior-address>  
3268 <btpt:superior-identifier>...URI...</btpt:superior-identifier>  
3269 <btpt:reply-address> ?  
3270 ...address...  
3271 </btpt:reply-address>  
3272 <btpt:superior-type>cohesion|atom</btpt:superior-type>  
3273 <btpt:qualifiers> ?  
3274 ...qualifiers...  
3275 </btpt:qualifiers>  
3276 </btpt:context>
```

3277

3278 CONTEXT_REPLY

3279

```
3280 <btpt:context-reply id?>  
3281 <btpt:target-additional-information> ?  
3282 ...additional address information...  
3283 </btpt:target-additional-information>  
3284  
3285 <btpt:superior-identifier>...URI...</btpt:superior-identifier>  
3286 <btpt:completion-  
3287 status>completed|related|repudiated</btpt:completion-status>  
3288 <btpt:qualifiers> ?  
3289 ...qualifiers...  
3290 </btpt:qualifiers>  
3291 </btpt:context-reply>
```

3292

3293
3294
3295
3296
3297
3298
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3340
3341
3342

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:fault-text>...string data ...</btpr:fault-data> ?  
  <btpr:qualifiers> ?  
    ...qualifiers...  
  </btpr:qualifiers>  
</btpr:fault>
```

3343 The following fault type names are represented by simple strings, corresponding to the entries
3344 defined in the abstract message set:

- 3345
- 3346 o communication-failure
- 3347 o duplicate-inferior
- 3348 o general
- 3349 o invalid-decider
- 3350 o invalid-inferior
- 3351 o invalid-superior
- 3352 o status-refused
- 3353 o invalid-terminator
- 3354 o unknown-parameter
- 3355 o unknown-transaction
- 3356 o unsupported-qualifier
- 3357 o wrong-state
- 3358 o [redirect](#)

3359
3360 Revisions of this specification may add other fault type names, which shall be simple strings
3361 of letters, numbers and hyphens. If other specifications define fault type names to be used
3362 with BTP, the names shall be URIs.

3363
3364 Fault data can take on various forms:

3365
3366 ~~Free text:~~

3367
3368 ~~<btp: fault-data>...string data...</btp: fault-data>~~

3369
3370 Identifier:

3371
3372 <btp: fault-data>...URI...</btp: fault-data>

3373
3374
3375 Inferior Identity:

3376
3377 <btp: fault-data>
3378 <btp: inferior-address> +
3379 ...address...
3380 </btp: inferior-address>
3381 <btp: inferior-identifier>...URI...</btp: inferior-identifier>
3382 </btp: fault-data>

3383
3384 ENROL

3385
3386 <btp: enrol id?>
3387 <btp: target-additional-information> ?
3388 ...additional address information...
3389 </btp: target-additional-information>
3390 <btp: superior-identifier>...URI...</btp: superior-identifier>

```
3391 <ctp:response-requested>true|false</ctp:response-requested>
3392 <ctp:reply-address> ?
3393   ...address...
3394 </ctp:reply-address>
3395 <ctp:inferior-address> +
3396   ...address...
3397 </ctp:inferior-address>
3398 <ctp:inferior-identifier>...URI...</ctp:inferior-identifier>
3399 <ctp:qualifiers> ?
3400   ...qualifiers...
3401 </ctp:qualifiers>
3402 </ctp:enrol>
```

3403
3404

ENROLLED

```
3405
3406
3407 <ctp:enrolled id?>
3408 <ctp:target-additional-information> ?
3409   ...additional address information...
3410 </ctp:target-additional-information>
3411 <ctp:sender-address> ?
3412   ...address...
3413 </ctp:sender-address>
3414 <ctp:inferior-identifier>...URI...</ctp:inferior-identifier>
3415 <ctp:qualifiers> ?
3416   ...qualifiers...
3417 </ctp:qualifiers>
3418 </ctp:enrolled>
```

3419
3420

RESIGN

```
3421
3422
3423 <ctp:resign id?>
3424 <ctp:target-additional-information> ?
3425   ...additional address information...
3426 </ctp:target-additional-information>
3427 <ctp:sender-address> ?
3428   ...address...
3429 </ctp:sender-address>
3430 <ctp:superior-identifier>...URI...</ctp:superior-identifier>
3431 <ctp:inferior-identifier>...URI...</ctp:inferior-identifier>
3432 <ctp:response-requested>true|false</ctp:response-requested>
3433 <ctp:qualifiers> ?
3434   ...qualifiers...
3435 </ctp:qualifiers>
3436 </ctp:resign>
```

3437
3438

RESIGNED

```
3439
3440
3441 <ctp:resigned id?>
```

```
3442 <btptarget-additional-information> ?
3443   ...additional address information...
3444 </btptarget-additional-information>
3445 <btptsender-address> ?
3446   ...address...
3447 </btptsender-address>
3448 <btptinferior-identifier>...URI...</btptinferior-identifier>
3449 <btptqualifiers> ?
3450   ...qualifiers...
3451 </btptqualifiers>
3452 </btptresigned>
```

PREPARE

```
3456 <btptprepare id?>
3457 <btpttarget-additional-information> ?
3458   ...additional address information...
3459 </btpttarget-additional-information>
3460 <btptsender-address> ?
3461   ...address...
3462 </btptsender-address>
3463 <btptinferior-identifier>...URI...</btptinferior-identifier>
3464 <btptqualifiers> ?
3465   ...qualifiers...
3466 </btptqualifiers>
3467 </btptprepare>
```

PREPARED

```
3471 <btptprepared id?>
3472 <btpttarget-additional-information> ?
3473   ...additional address information...
3474 </btpttarget-additional-information>
3475 <btptsender-address> ?
3476   ...address...
3477 </btptsender-address>
3478 <btptsuperior-identifier>...URI...</btptsuperior-identifier>
3479 <btptinferior-identifier>...URI...</btptinferior-identifier>
3480 <btptdefault-is-cancel>true|false</btptdefault-is-cancel>
3481 <btptqualifiers> ?
3482   ...qualifiers...
3483 </btptqualifiers>
3484 </btptprepared>
```

CONFIRM

```
3488 <btptconfirm id?>
3489 <btpttarget-additional-information> ?
```

```
3493     ...additional address information...
3494     </btp:target-additional-information>
3495     <btp:sender-address> ?
3496     ...address...
3497     </btp:sender-address>
3498     <btp:inferior-identifier>...URI...</btp:inferior-identifier>
3499     <btp:qualifiers> ?
3500     ...qualifiers...
3501     </btp:qualifiers>
3502 </btp:confirm>
```

CONFIRMED

```
3503
3504
3505
3506
3507     <btp:confirmed id?>
3508     <btp:target-additional-information> ?
3509     ...additional address information...
3510     </btp:target-additional-information>
3511     <btp:sender-address> ?
3512     ...address...
3513     </btp:sender-address>
3514     <btp:superior-identifier>...URI...</btp:superior-identifier>
3515     <btp:inferior-identifier>...URI...</btp:inferior-identifier>
3516     <btp:confirmed-received>true|false</btp:confirmed-received>
3517     <btp:qualifiers> ?
3518     ...qualifiers...
3519     </btp:qualifiers>
3520 </btp:confirmed>
```

CANCEL

```
3521
3522
3523
3524
3525     <btp:cancel id?>
3526     <btp:target-additional-information> ?
3527     ...additional address information...
3528     </btp:target-additional-information>
3529     <btp:sender-address> ?
3530     ...address...
3531     </btp:sender-address>
3532     <btp:inferior-identifier>...URI...</btp:inferior-identifier>
3533     <btp:reply-address> ?
3534     ...address...
3535     </btp:reply-address>
3536     <btp:qualifiers> ?
3537     ...qualifiers...
3538     </btp:qualifiers>
3539 </btp:cancel>
```

CANCELLED

3540
3541
3542
3543

```
3544 <btpt:cancelled id?>
3545   <btpt:target-additional-information> ?
3546     ...additional address information...
3547   </btpt:target-additional-information>
3548   <btpt:sender-address> ?
3549     ...address...
3550   </btpt:sender-address>
3551   <btpt:superior-identifier>...URI...</btpt:superior-identifier>
3552
3553   <btpt:inferior-identifier>...URI...</btpt:inferior-identifier> ?
3554   <btpt:qualifiers> ?
3555     ...qualifiers...
3556   </btpt:qualifiers>
3557 </btpt:cancelled>
```

3558

3559

CONFIRM_ONE_PHASE

3560

3561

3562

3563

3564

3565

3566

3567

3568

3569

3570

3571

3572

3573

3574

3575

```
<btpt:confirm-one-phase id?>
  <btpt:target-additional-information> ?
    ...additional address information...
  </btpt:target-additional-information>
  <btpt:sender-address> ?
    ...address...
  </btpt:sender-address>
  <btpt:inferior-identifier>...URI...</btpt:inferior-identifier>
  <btpt:report-hazard>true|false</btpt:report-hazard>
  <btpt:qualifiers> ?
    ...qualifiers...
  </btpt:qualifiers>
</btpt:confirm-one-phase>
```

3576

HAZARD

3577

3578

3579

3580

3581

3582

3583

3584

3585

3586

3587

3588

3589

3590

3591

3592

3593

3594

```
<btpt:hazard id?>
  <btpt:target-additional-information> ?
    ...additional address information...
  </btpt:target-additional-information>
  <btpt:sender-address> ?
    ...address...
  </btpt:sender-address>
  <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>
```

3595
3596
3597
3598
3599
3600
3601
3602
3603
3604
3605
3606
3607
3608
3609
3610

CONTRADICTION

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

3611
3612
3613
3614
3615
3616
3617
3618
3619
3620
3621
3622
3623
3624
3625
3626
3627
3628
3629

SUPERIOR_STATE

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

3630
3631
3632
3633
3634
3635
3636
3637
3638
3639
3640
3641
3642
3643
3644
3645
3646

INFERIOR_STATE

```
<btpr:inferior-state id?>  
  <btpr:target-additional-information> ?  
    ...additional address information...  
</btpr:target-additional-information>  
  <btpr:sender-address> ?  
    ...address...  
</btpr:sender-address>  
<btpr:superior-identifier>...URI...</btpr:superior-identifier>  
  
<btpr:inferior-identifier>...URI...</btpr:inferior-identifier>  
<btpr:status>active|inaccessible|unknown</btpr:status>  
<btpr:response-requested>true|false</btpr:response-requested>  
<btpr:qualifiers> ?  
  ...qualifiers...  
</btpr:qualifiers>
```

3647 </btp:inferior-state>

3648

3649

3650

REDIRECT

3651

3652

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

3653

```
<btp:target-additional-information> ?
```

3654

```
...additional address information...
```

3655

```
</btp:target-additional-information>
```

3656

```
<btp:superior-identifier>...URI...</btp:superior-identifier> ?
```

3657

```
<btp:inferior-identifier>...URI...</btp:inferior-identifier>
```

3658

```
<btp:old-address> +
```

3659

```
...address...
```

3660

```
</btp:old-address>
```

3661

```
<btp:new-address> +
```

3662

```
...address...
```

3663

```
</btp:new-address>
```

3664

```
<btp:qualifiers> ?
```

3665

```
...qualifiers...
```

3666

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

3667

```
</btp:redirect>
```

3668

3669

BEGIN

3670

```
<btp:begin id?>
```

3672

```
<btp:target-additional-information> ?
```

3673

```
...additional address information...
```

3674

```
</btp:target-additional-information>
```

3675

```
<btp:reply-address> ?
```

3676

```
...address...
```

3677

```
</btp:reply-address>
```

3678

```
<btp:transaction-type>cohesion|atom</btp:transaction-type>
```

3679

```
<btp:qualifiers> ?
```

3680

```
...qualifiers...
```

3681

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

3682

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

3683

3684

3685

BEGUN

3686

```
<btp:begun id?>
```

3688

```
<btp:target-additional-information> ?
```

3689

```
...additional address information...
```

3690

```
</btp:target-additional-information>
```

3691

```
<btp:decider-address> *
```

3692

```
...address...
```

3693

```
</btp:decider-address>
```

3694

```
<btp:inferior-address> *
```

3695

```
...address...
```

3696

```
</btp:inferior-address>
```

```
3697     <btpt:transaction-identifier>...URI...</btpt:transaction-
3698 identifier>
3699     <btpt:qualifiers> ?
3700     ...qualifiers...
3701     </btpt:qualifiers>
3702 </btpt:begin>
```

PREPARE_INFERIORS

```
3707 <btpt:prepare-inferiors id?>
3708   <btpt:target-additional-information> ?
3709   ...additional address information...
3710 </btpt:target-additional-information>
3711   <btpt:reply-address> ?
3712   ...address...
3713 </btpt:reply-address>
3714   <btpt:transaction-identifier>...URI...</btpt:transaction-
3715 identifier>
3716   <btpt:inferiors-list> ?
3717     <btpt:inferior-handle>...URI...</btpt:inferior-handle> +
3718 </btpt:inferiors-list>
3719   <btpt:qualifiers> ?
3720   ...qualifiers...
3721   </btpt:qualifiers>
3722 </btpt:prepare-inferiors>
```

CONFIRM_TRANSACTION

```
3727 <btpt:confirm-transaction id?>
3728   <btpt:target-additional-information> ?
3729   ...additional address information...
3730 </btpt:target-additional-information>
3731   <btpt:reply-address> ?
3732   ...address...
3733 </btpt:reply-address>
3734   <btpt:transaction-identifier>...URI...</btpt:transaction-
3735 identifier>
3736   <btpt:inferiors-list> ?
3737     <btpt:inferior-handle>...URI...</btpt:inferior-handle> +
3738 </btpt:inferiors-list>
3739   <btpt:report-hazard>true|false</btpt:report-hazard>
3740   <btpt:qualifiers> ?
3741   ...qualifiers...
3742   </btpt:qualifiers>
3743 </btpt:confirm_transaction>
```

TRANSACTION_CONFIRMED

3747

```
3748 <btpt:transaction-confirmed id?>
3749   <btpt:target-additional-information> ?
3750     ...additional address information...
3751   </btpt:target-additional-information>
3752
3753   <btpt:transaction-identifier>...URI...</btpt:transaction-
3754   identifier>
3755   <btpt:qualifiers> ?
3756     ...qualifiers...
3757   </btpt:qualifiers>
3758 </btpt:transaction-confirmed>
```

3759

3760

CANCEL_TRANSACTION

3761

3762

```
3763 <btpt:cancel-transaction id?>
3764   <btpt:target-additional-information> ?
3765     ...additional address information...
3766   </btpt:target-additional-information>
3767   <btpt:reply-address> ?
3768     ...address...
3769   </btpt:reply-address>
3770   <btpt:transaction-identifier>...URI...</btpt:transaction-
3771   identifier>
3772   <btpt:report-hazard>true|false</btpt:report-hazard>
3773   <btpt:qualifiers> ?
3774     ...qualifiers...
3775   </btpt:qualifiers>
3776 </btpt:cancel-transaction>
```

3777

3778

CANCEL_INFERIORS

3779

```
3780 <btpt:cancel-inferiors id?>
3781   <btpt:target-additional-information> ?
3782     ...additional address information...
3783   </btpt:target-additional-information>
3784   <btpt:reply-address> ?
3785     ...address...
3786   </btpt:reply-address>
3787   <btpt:transaction-identifier>...URI...</btpt:transaction-
3788   identifier> ?
3789   <btpt:inferiors-list>
3790     <btpt:inferior-handle>...URI...</btpt:inferior-handle> +
3791   </btpt:inferiors-list>
3792   <btpt:qualifiers> ?
3793     ...qualifiers...
3794   </btpt:qualifiers>
3795 </btpt:cancel-inferiors>
```

3796

3797

3798
3799
3800
3801
3802
3803
3804
3805
3806
3807
3808
3809
3810
3811
3812

TRANSACTION_CANCELLED

```
<btpt:transaction-cancelled id?>
  <btpt:target-additional-information> ?
  ...additional address information...
</btpt:target-additional-information>

  <btpt:transaction-identifier>...URI...</btpt:transaction-
identifier>
  <btpt:qualifiers> ?
  ...qualifiers...
</btpt:qualifiers>
</btpt:transaction-cancelled>
```

3813
3814
3815
3816
3817
3818
3819
3820
3821
3822
3823
3824
3825
3826
3827
3828
3829
3830
3831

REQUEST_INFERIOR_STATUSES

```
<btpt:request-inferior-statuses id?>
  <btpt:target-additional-information> ?
  ...additional address information...
</btpt:target-additional-information>
  <btpt:reply-address> ?
  ...address...
</btpt:reply-address>
  <btpt:target-identifier>...URI...</btpt:target-identifier>
  <btpt:inferiors-list> ?
  <btpt:inferior-handle>...URI...</btpt:inferior-handle> +
</btpt:inferiors-list>
  <btpt:qualifiers> ?
  ...qualifiers...
</btpt:qualifiers>
</btpt:request-inferior-statuses>
```

3832

INFERIOR_STATUSES

3833
3834
3835
3836
3837
3838
3839
3840
3841
3842
3843
3844
3845
3846
3847
3848
3849

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

  <btpt:responders-identifier>...URI...</btpt:responders-identifier>
  <btpt:status-list>
  <btpt:status-item> +
  <btpt:inferior-handle>...URI...</btpt:inferior-handle>
  <btpt:status>active|resigned|preparing|prepared|
  autonomously-confirmed|autonomously-cancelled|
  confirming|confirmed|cancelling|cancelled|
  cancel-contradiction|confirm-contradiction|
  hazard|invalid</btpt:status>
  <btpt:qualifiers> ?
  ...qualifiers...
```

```
3850         </btp:qualifiers>
3851     </btp:status-item>
3852 </btp:status-list>
3853 <btp:qualifiers> ?
3854     ...qualifiers...
3855 </btp:qualifiers>
3856 </btp:inferior-statuses>
3857
```

3858 **Standard qualifiers**

3859 The informal syntax for these messages assumes the namespace prefix “btpq” is associated
3860 with the URI “urn:oasis:names:tc:BTP:[1.0:qualifiers](#)”.

3862 **Transaction timelimit**

```
3863
3864 <btpq:transaction-timelimit>
3865 <btpq:timelimit>
3866     ...time in seconds...
3867 </btpq:timelimit>
3868 </btpq:transaction-timelimit>
3869
```

3870 **Inferior timeout**

```
3871 <btpq:inferior-timeout>
3872 <btpq:timeout>
3873     ...time in seconds...
3874 </btpq:timeout>
3875 <btpq:intended-decision>confirm|cancel</btpq:intended-decision>
3876 </btpq:inferior-timeout>
3877
```

3878 **Minimum inferior timeout**

```
3879 <btpq:minimum-inferior-timeout>
3880 <btpq:minimum-timeout>
3881     ...time in seconds...
3882 </btpq:minimum-timeout>
3883 </btpq:minimum-inferior-timeout>
3884
```

3885 **Inferior name**

```
3886 <btpq:inferior-name>
3887 <btpq:inferior-name>
3888     ...string...
3889 </btpq:inferior-name>
3890 </btpq:inferior-name>
3891
```

3892 **Compounding of Messages**

3893
3894 Relating BTP to one another, in a “group” is represented by containing them within the
3895 btp:related-group element, with the related messages as child elements. The processing for
3896 the group is defined in the section “Groups – combinations of related messages”. For example

```
3897
3898 <btp:related-group>
3899     <btp:context-reply>
```

```
3900     ...<completion-status>related</completion-status> ...
3901     </btp:context-reply>
3902     <btp:enrol>...</btp:enrol>
3903     <btp:prepared>...</btp:prepared>
3904 </btp:related-group>
```

3906 If the rules for the group state that the “target-address” of the abstract message is omitted, the
3907 corresponding target-address-information element shall be absent in the message in the
3908 related-group. The carrier protocol binding specifies how a relation between application and
3909 BTP messages is represented.

3911 Bundling (semantically insignificant combination) of BTP messages and related groups is
3912 indicated with the "btp:messages" element, with the bundled messages and related groups as
3913 child elements. For example (confirming one and cancelling another inferiors of a cohesion):

```
3914 <btp:messages>
3915   <btp:confirm>...</btp:confirm>
3916   <btp:cancel>...</btp:cancel>
3917 </btp:messages>
```

3919
3920
3921

3921

XML Schemas

3922

3923

3924

XML schema for BTP messages

3925

3926

3927

3928

3929

3930

3931

3932

3933

3934

3935

3936

3937

3938

3939

3940

3941

3942

3943

3944

3945

3946

3947

3948

3949

3950

3951

3952

3953

3954

3955

3956

3957

3958

3959

3960

3961

3962

3963

3964

3965

3966

3967

3968

3969

3970

3971

3972

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

  <!-- Qualifiers -->

  <complexType name="qualifier-type">
    <simpleContent>
      <extension base="string">
        <attribute name="must-be-understood" type="boolean"/>
        <attribute name="to-be-propagated" type="boolean"/>
      </extension>
    </simpleContent>
  </complexType>

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

  <element name="qualifiers">
    <complexType>
      <sequence>
        <element ref="btp:qualifier" maxOccurs="unbounded"/>
      </sequence>
    </complexType>
  </element>

  <!-- example qualifier:
  <element name="some-qualifer" type="btp:qualifier-type"
substitutionGroup="btp:qualifier"/>
  -->

  <!-- Message set data types -->

  <simpleType name="identifier">
    <restriction base="anyURI" />
  </simpleType>

  <simpleType name="additional-information">
    <restriction base="string" />
  </simpleType>

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

```

3973         <element name="binding-name" type="anyURI"/>
3974         <element name="binding-address" type="string"/>
3975         <element name="additional-information" type="btp:additional-
3976 information" minOccurs="0" />
3977     </sequence>
3978 </complexType>
3979
3980 <simpleType name="superior-type">
3981     <restriction base="string">
3982         <enumeration value="cohesion"/>
3983         <enumeration value="atom"/>
3984     </restriction>
3985 </simpleType>
3986
3987 <simpleType name="transaction-type">
3988     <restriction base="string">
3989         <enumeration value="cohesion"/>
3990         <enumeration value="atom"/>
3991     </restriction>
3992 </simpleType>
3993
3994
3995 <!-- Compounding -->
3996
3997 <element name="messages">
3998     <complexType>
3999         <sequence>
4000             <element ref="btp:message" minOccurs="0"
4001 maxOccurs="unbounded"/>
4002         </sequence>
4003     </complexType>
4004 </element>
4005
4006 <element name="related-group" substitutionGroup="btp:message">
4007     <complexType>
4008         <sequence>
4009             <element ref="btp:message" minOccurs="0"
4010 maxOccurs="unbounded"/>
4011         </sequence>
4012     </complexType>
4013 </element>
4014
4015
4016 <!-- Message set -->
4017
4018 <element name="message" abstract="true" />
4019
4020 <element name="context" substitutionGroup="btp:message">
4021     <complexType>
4022         <sequence>
4023             <element name="superior-address" type="btp:address"
4024 maxOccurs="unbounded"/>
4025             <element name="superior-identifier" type="btp:identifier"/>

```

```

4026         <element name="reply-address" type="btp:address"
4027 minOccurs="0"/>
4028         <element name="superior-type" type="btp:superior-type"/>
4029         <element ref="btp:qualifiers" minOccurs="0"/>
4030     </sequence>
4031     <attribute name="id" type="ID" use="optional"/>
4032 </complexType>
4033 </element>
4034
4035     <element name="context-reply" substitutionGroup="btp:message">
4036         <complexType>
4037             <sequence>
4038                 <element name="target-additional-information"
4039 type="btp:additional-information" minOccurs="0"/>
4040                 <element name="superior-identifier" type="btp:identifier"/>
4041                 <element name="completion-status">
4042                     <simpleType>
4043                         <restriction base="string">
4044                             <enumeration value="completed"/>
4045                             <enumeration value="related"/>
4046                             <enumeration value="repudiated"/>
4047                         </restriction>
4048                     </simpleType>
4049                 </element>
4050                 <element ref="btp:qualifiers" minOccurs="0"/>
4051             </sequence>
4052             <attribute name="id" type="ID"/>
4053         </complexType>
4054     </element>
4055
4056     <element name="request-status" substitutionGroup="btp:message">
4057         <complexType>
4058             <sequence>
4059                 <element name="target-additional-information"
4060 type="btp:additional-information" minOccurs="0"/>
4061                 <element name="reply-address" type="btp:address"
4062 minOccurs="0"/>
4063                 <element name="target-identifier" type="btp:identifier"/>
4064                 <element ref="btp:qualifiers" minOccurs="0"/>
4065             </sequence>
4066             <attribute name="id" type="ID"/>
4067         </complexType>
4068     </element>
4069
4070     <element name="status" substitutionGroup="btp:message">
4071         <complexType>
4072             <sequence>
4073                 <element name="target-additional-information"
4074 type="btp:additional-information" minOccurs="0"/>
4075                 <element name="responders-identifier"
4076 type="btp:identifier"/>
4077                 <element name="status-value">
4078                     <simpleType>

```

```

4079         <restriction base="string">
4080             <enumeration value="created"/>
4081             <enumeration value="enrolling"/>
4082             <enumeration value="active"/>
4083             <enumeration value="resigning"/>
4084             <enumeration value="resigned"/>
4085             <enumeration value="preparing"/>
4086             <enumeration value="prepared"/>
4087             <enumeration value="confirming"/>
4088             <enumeration value="confirmed"/>
4089             <enumeration value="cancelling"/>
4090             <enumeration value="cancelled"/>
4091             <enumeration value="cancel-contradiction"/>
4092             <enumeration value="confirm-contradiction"/>
4093             <enumeration value="hazard"/>
4094             <enumeration value="contradicted"/>
4095             <enumeration value="unknown"/>
4096             <enumeration value="inaccessible"/>
4097         </restriction>
4098     </simpleType>
4099 </element>
4100     <element ref="btp:qualifiers" minOccurs="0"/>
4101 </sequence>
4102     <attribute name="id" type="ID"/>
4103 </complexType>
4104 </element>
4105
4106     <element name="fault" substitutionGroup="btp:message">
4107         <complexType>
4108             <sequence>
4109                 <element name="target-additional-information"
4110 type="btp:additional-information" minOccurs="0"/>
4111                 <element name="superior-identifier" type="btp:identifier"
4112 minOccurs="0"/>
4113                 <element name="inferior-identifier" type="btp:identifier"
4114 minOccurs="0"/>
4115                 <element name="fault-type">
4116                     <simpleType>
4117                         <restriction base="string">
4118                             <enumeration value="communication-failure"/>
4119                             <enumeration value="duplicate-inferior"/>
4120                             <enumeration value="general"/>
4121                             <enumeration value="invalid-decider"/>
4122                             <enumeration value="invalid-inferior"/>
4123                             <enumeration value="invalid-superior"/>
4124                             <enumeration value="status-refused"/>
4125                             <enumeration value="invalid-terminator"/>
4126                             <enumeration value="unknown-parameter"/>
4127                             <enumeration value="unknown-transaction"/>
4128                             <enumeration value="unsupported-qualifier"/>
4129                             <enumeration value="wrong-state"/>
4130                         </restriction>
4131                     </simpleType>

```

```

4132         </element>
4133         <element name="fault-data" type="anyType" minOccurs="0"/>
4134         <element ref="btp:qualifiers" minOccurs="0"/>
4135     </sequence>
4136     <attribute name="id" type="ID"/>
4137 </complexType>
4138 </element>
4139
4140 <element name="enrol" substitutionGroup="btp:message">
4141     <complexType>
4142         <sequence>
4143             <element name="target-additional-information"
4144 type="btp:additional-information" minOccurs="0"/>
4145             <element name="superior-identifier" type="btp:identifier"/>
4146             <element name="response-requested" type="boolean"/>
4147             <element name="reply-address" type="btp:address"
4148 minOccurs="0"/>
4149             <element name="inferior-address" type="btp:address"
4150 minOccurs="1" maxOccurs="unbounded"/>
4151             <element name="inferior-identifier" type="btp:identifier"/>
4152             <element ref="btp:qualifiers" minOccurs="0"/>
4153         </sequence>
4154         <attribute name="id" type="ID"/>
4155     </complexType>
4156 </element>
4157
4158
4159 <element name="enrolled" substitutionGroup="btp:message">
4160     <complexType>
4161         <sequence>
4162             <element name="target-additional-information"
4163 type="btp:additional-information" minOccurs="0"/>
4164             <element name="inferior-identifier" type="btp:identifier"/>
4165             <element ref="btp:qualifiers" minOccurs="0"/>
4166         </sequence>
4167         <attribute name="id" type="ID"/>
4168     </complexType>
4169 </element>
4170
4171 <element name="resign" substitutionGroup="btp:message">
4172     <complexType>
4173         <sequence>
4174             <element name="target-additional-information"
4175 type="btp:additional-information" minOccurs="0"/>
4176             <element name="superior-identifier" type="btp:identifier"/>
4177             <element name="inferior-identifier" type="btp:identifier"/>
4178             <element name="response-requested" type="boolean"/>
4179             <element ref="btp:qualifiers" minOccurs="0"/>
4180         </sequence>
4181         <attribute name="id" type="ID"/>
4182     </complexType>
4183 </element>
4184

```

```

4185     <element name="resigned" substitutionGroup="btp:message">
4186         <complexType>
4187             <sequence>
4188                 <element name="target-additional-information"
4189 type="btp:additional-information" minOccurs="0"/>
4190                 <element name="inferior-identifier" type="btp:identifier"/>
4191                 <element ref="btp:qualifiers" minOccurs="0"/>
4192             </sequence>
4193             <attribute name="id" type="ID"/>
4194         </complexType>
4195     </element>
4196
4197     <element name="prepare" substitutionGroup="btp:message">
4198         <complexType>
4199             <sequence>
4200                 <element name="target-additional-information"
4201 type="btp:additional-information" minOccurs="0"/>
4202                 <element name="inferior-identifier" type="btp:identifier"/>
4203                 <element ref="btp:qualifiers" minOccurs="0"/>
4204             </sequence>
4205             <attribute name="id" type="ID"/>
4206         </complexType>
4207     </element>
4208
4209     <element name="prepared" 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                 <element name="inferior-identifier" type="btp:identifier"/>
4216                 <element name="default-is-cancel" type="boolean"/>
4217                 <element ref="btp:qualifiers" minOccurs="0"/>
4218             </sequence>
4219             <attribute name="id" type="ID"/>
4220         </complexType>
4221     </element>
4222
4223     <element name="confirm" substitutionGroup="btp:message">
4224         <complexType>
4225             <sequence>
4226                 <element name="target-additional-information"
4227 type="btp:additional-information" minOccurs="0"/>
4228                 <element name="inferior-identifier" type="btp:identifier"/>
4229                 <element ref="btp:qualifiers" minOccurs="0"/>
4230             </sequence>
4231             <attribute name="id" type="ID"/>
4232         </complexType>
4233     </element>
4234
4235     <element name="confirmed" substitutionGroup="btp:message">
4236         <complexType>
4237             <sequence>

```

```

4238         <element name="target-additional-information"
4239 type="btp:additional-information" minOccurs="0"/>
4240         <element name="superior-identifier" type="btp:identifier"/>
4241         <element name="inferior-identifier" type="btp:identifier"/>
4242         <element name="confirmed-received" type="boolean"/>
4243         <element ref="btp:qualifiers" minOccurs="0"/>
4244     </sequence>
4245     <attribute name="id" type="ID"/>
4246 </complexType>
4247 </element>
4248
4249 <element name="cancel" substitutionGroup="btp:message">
4250 <complexType>
4251 <sequence>
4252     <element name="target-additional-information"
4253 type="btp:additional-information" minOccurs="0"/>
4254     <element name="inferior-identifier" type="btp:identifier"/>
4255     <element name="reply-address" type="btp:address"
4256 minOccurs="0"/>
4257     <element ref="btp:qualifiers" minOccurs="0"/>
4258 </sequence>
4259 <attribute name="id" type="ID"/>
4260 </complexType>
4261 </element>
4262
4263 <element name="cancelled" substitutionGroup="btp:message">
4264 <complexType>
4265 <sequence>
4266     <element name="target-additional-information"
4267 type="btp:additional-information" minOccurs="0"/>
4268     <element name="superior-identifier" type="btp:identifier"/>
4269     <element name="inferior-identifier" type="btp:identifier"
4270 minOccurs="0"/>
4271     <element ref="btp:qualifiers" minOccurs="0"/>
4272 </sequence>
4273 <attribute name="id" type="ID"/>
4274 </complexType>
4275 </element>
4276
4277 <element name="confirm-one-phase" substitutionGroup="btp:message">
4278 <complexType>
4279 <sequence>
4280     <element name="target-additional-information"
4281 type="btp:additional-information" minOccurs="0"/>
4282     <element name="inferior-identifier" type="btp:identifier"/>
4283     <element name="report-hazard" type="boolean"/>
4284     <element ref="btp:qualifiers" minOccurs="0"/>
4285 </sequence>
4286 <attribute name="id" type="ID"/>
4287 </complexType>
4288 </element>
4289
4290 <element name="hazard" substitutionGroup="btp:message">

```

```

4291     <complexType>
4292         <sequence>
4293             <element name="target-additional-information"
4294 type="btp:additional-information" minOccurs="0"/>
4295             <element name="superior-identifier" type="btp:identifier"/>
4296             <element name="inferior-identifier" type="btp:identifier"/>
4297             <element name="level">
4298                 <simpleType>
4299                     <restriction base="string">
4300                         <enumeration value="mixed"/>
4301                         <enumeration value="possible"/>
4302                     </restriction>
4303                 </simpleType>
4304             </element>
4305             <element ref="btp:qualifiers" minOccurs="0"/>
4306         </sequence>
4307         <attribute name="id" type="ID"/>
4308     </complexType>
4309 </element>
4310
4311 <element name="contradiction" substitutionGroup="btp:message">
4312     <complexType>
4313         <sequence>
4314             <element name="target-additional-information"
4315 type="btp:additional-information" minOccurs="0"/>
4316             <element name="inferior-identifier" type="btp:identifier"/>
4317             <element ref="btp:qualifiers" minOccurs="0"/>
4318         </sequence>
4319         <attribute name="id" type="ID"/>
4320     </complexType>
4321 </element>
4322
4323 <element name="superior-state" substitutionGroup="btp:message">
4324     <complexType>
4325         <sequence>
4326             <element name="target-additional-information"
4327 type="btp:additional-information" minOccurs="0"/>
4328             <element name="inferior-identifier" type="btp:identifier"/>
4329             <element name="status">
4330                 <simpleType>
4331                     <restriction base="string">
4332                         <enumeration value="active"/>
4333                         <enumeration value="prepared-received"/>
4334                         <enumeration value="inaccessible"/>
4335                         <enumeration value="unknown"/>
4336                     </restriction>
4337                 </simpleType>
4338             </element>
4339             <element name="response-requested" type="boolean"/>
4340             <element ref="btp:qualifiers" minOccurs="0"/>
4341         </sequence>
4342         <attribute name="id" type="ID"/>
4343     </complexType>

```

```

4344     </element>
4345
4346     <element name="inferior-state" substitutionGroup="btp:message">
4347         <complexType>
4348             <sequence>
4349                 <element name="target-additional-information"
4350 type="btp:additional-information" minOccurs="0"/>
4351                 <element name="superior-identifier" type="btp:identifier"/>
4352                 <element name="inferior-identifier" type="btp:identifier"/>
4353                 <element name="status">
4354                     <simpleType>
4355                         <restriction base="string">
4356                             <enumeration value="active"/>
4357                             <enumeration value="inaccessible"/>
4358                             <enumeration value="unknown"/>
4359                         </restriction>
4360                     </simpleType>
4361                 </element>
4362                 <element name="response-requested" type="boolean"/>
4363                 <element ref="btp:qualifiers" minOccurs="0"/>
4364             </sequence>
4365             <attribute name="id" type="ID"/>
4366         </complexType>
4367     </element>
4368
4369     <element name="redirect" substitutionGroup="btp:message">
4370         <complexType>
4371             <sequence>
4372                 <element name="target-additional-information"
4373 type="btp:additional-information" minOccurs="0"/>
4374                 <element name="superior-identifier" type="btp:identifier"
4375 minOccurs="0"/>
4376                 <element name="inferior-identifier" type="btp:identifier"
4377 />
4378                 <element name="old-address" type="btp:address"
4379 maxOccurs="unbounded"/>
4380                 <element name="new-address" type="btp:address"
4381 maxOccurs="unbounded"/>
4382                 <element ref="btp:qualifiers" minOccurs="0"/>
4383             </sequence>
4384             <attribute name="id" type="ID"/>
4385         </complexType>
4386     </element>
4387
4388
4389     <element name="begin" substitutionGroup="btp:message">
4390         <complexType>
4391             <sequence>
4392                 <element name="target-additional-information"
4393 type="btp:additional-information" minOccurs="0"/>
4394                 <element name="reply-address" type="btp:address"
4395 minOccurs="0"/>
4396                 <element name="transaction-type" type="btp:superior-type"/>

```

```

4397         <element ref="btp:qualifiers" minOccurs="0"/>
4398     </sequence>
4399     <attribute name="id" type="ID"/>
4400 </complexType>
4401 </element>
4402
4403     <element name="begun" substitutionGroup="btp:message">
4404         <complexType>
4405             <sequence>
4406                 <element name="target-additional-information"
4407 type="btp:additional-information" minOccurs="0"/>
4408                 <element name="decider-address" type="btp:address"
4409 minOccurs="0" maxOccurs="unbounded"/>
4410                 <element name="transaction-identifier"
4411 type="btp:identifier" minOccurs="0"/>
4412                 <element name="inferior-handle" type="btp:identifier"
4413 minOccurs="0"/>
4414                 <element name="inferior-address" type="btp:address"
4415 minOccurs="0" maxOccurs="unbounded"/>
4416                 <element ref="btp:qualifiers" minOccurs="0"/>
4417             </sequence>
4418             <attribute name="id" type="ID"/>
4419         </complexType>
4420     </element>
4421
4422     <element name="prepare-inferiors" substitutionGroup="btp:message">
4423         <complexType>
4424             <sequence>
4425                 <element name="target-additional-information"
4426 type="btp:additional-information" minOccurs="0"/>
4427                 <element name="reply-address" type="btp:address"
4428 minOccurs="0"/>
4429                 <element name="transaction-identifier"
4430 type="btp:identifier"/>
4431                 <element name="inferiors-list" minOccurs="0">
4432                     <complexType>
4433                         <sequence>
4434                             <element name="inferior-handle"
4435 type="btp:identifier" maxOccurs="unbounded"/>
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     <element name="confirm-transaction" substitutionGroup="btp:message">
4446         <complexType>
4447             <sequence>
4448                 <element name="target-additional-information"
4449 type="btp:additional-information" minOccurs="0"/>

```

```

4450         <element name="reply-address" type="btp:address"
4451 minOccurs="0"/>
4452         <element name="transaction-identifier"
4453 type="btp:identifier"/>
4454         <element name="inferiors-list" minOccurs="0">
4455             <complexType>
4456                 <sequence>
4457                     <element name="inferior-handle"
4458 type="btp:identifier" maxOccurs="unbounded"/>
4459                 </sequence>
4460             </complexType>
4461         </element>
4462         <element name="report-hazard" type="boolean"/>
4463         <element ref="btp:qualifiers" minOccurs="0"/>
4464     </sequence>
4465     <attribute name="id" type="ID"/>
4466 </complexType>
4467 </element>
4468
4469     <element name="transaction-confirmed" substitutionGroup="btp:message">
4470         <complexType>
4471             <sequence>
4472                 <element name="target-additional-information"
4473 type="btp:additional-information" minOccurs="0"/>
4474                 <element name="transaction-identifier"
4475 type="btp:identifier"/>
4476                 <element ref="btp:qualifiers" minOccurs="0"/>
4477             </sequence>
4478             <attribute name="id" type="ID"/>
4479         </complexType>
4480     </element>
4481
4482     <element name="cancel-transaction" substitutionGroup="btp:message">
4483         <complexType>
4484             <sequence>
4485                 <element name="target-additional-information"
4486 type="btp:additional-information" minOccurs="0"/>
4487                 <element name="reply-address" type="btp:address"
4488 minOccurs="0"/>
4489                 <element name="transaction-identifier"
4490 type="btp:identifier"/>
4491                 <element name="report-hazard" type="boolean"/>
4492                 <element ref="btp:qualifiers" minOccurs="0"/>
4493             </sequence>
4494             <attribute name="id" type="ID"/>
4495         </complexType>
4496     </element>
4497
4498     <element name="cancel-inferiors" substitutionGroup="btp:message">
4499         <complexType>
4500             <sequence>
4501                 <element name="target-additional-information"
4502 type="btp:additional-information" minOccurs="0"/>

```

```

4503         <element name="reply-address" type="btp:address"
4504 minOccurs="0"/>
4505         <element name="transaction-identifier"
4506 type="btp:identifier" minOccurs="0"/>
4507         <element name="inferiors-list">
4508             <complexType>
4509                 <sequence>
4510                     <element name="inferior-handle"
4511 type="btp:identifier" maxOccurs="unbounded"/>
4512                 </sequence>
4513             </complexType>
4514         </element>
4515         <element ref="btp:qualifiers" minOccurs="0"/>
4516     </sequence>
4517     <attribute name="id" type="ID"/>
4518 </complexType>
4519 </element>
4520
4521     <element name="transaction-cancelled" substitutionGroup="btp:message">
4522         <complexType>
4523             <sequence>
4524                 <element name="target-additional-information"
4525 type="btp:additional-information" minOccurs="0"/>
4526                 <element name="transaction-identifier"
4527 type="btp:identifier"/>
4528                 <element ref="btp:qualifiers" minOccurs="0"/>
4529             </sequence>
4530             <attribute name="id" type="ID"/>
4531         </complexType>
4532     </element>
4533
4534     <element name="request-inferior-statuses"
4535 substitutionGroup="btp:message">
4536         <complexType>
4537             <sequence>
4538                 <element name="target-additional-information"
4539 type="btp:additional-information" minOccurs="0"/>
4540                 <element name="reply-address" type="btp:address"
4541 minOccurs="0"/>
4542                 <element name="target-identifier" type="btp:identifier"/>
4543                 <element name="inferiors-list" minOccurs="0">
4544                     <complexType>
4545                         <sequence>
4546                             <element name="inferior-handle"
4547 type="btp:identifier" maxOccurs="unbounded"/>
4548                         </sequence>
4549                     </complexType>
4550                 </element>
4551                 <element ref="btp:qualifiers" minOccurs="0"/>
4552             </sequence>
4553             <attribute name="id" type="ID"/>
4554         </complexType>
4555     </element>

```

```

4556
4557     <element name="inferior-statuses" substitutionGroup="btp:message">
4558         <complexType>
4559             <sequence>
4560                 <element name="target-additional-information"
4561 type="btp:additional-information" minOccurs="0"/>
4562                 <element name="responders-identifier"
4563 type="btp:identifier"/>
4564                 <element name="status-list">
4565                     <complexType>
4566                         <sequence>
4567                             <element name="status-item" maxOccurs="unbounded">
4568                                 <complexType>
4569                                     <sequence>
4570                                         <element name="inferior-handle"
4571 type="btp:identifier"/>
4572                                         <element name="status">
4573                                             <simpleType>
4574                                                 <restriction base="string">
4575                                                     <enumeration value="active"/>
4576                                                     <enumeration value="resigned"/>
4577                                                     <enumeration value="preparing"/>
4578                                                     <enumeration value="prepared"/>
4579                                                     <enumeration value="autonomously-confirmed"/>
4580                                                     <enumeration value="autonomously-cancelled"/>
4581                                                     <enumeration value="confirming"/>
4582                                                     <enumeration value="confirmed"/>
4583                                                     <enumeration value="cancelling"/>
4584                                                     <enumeration value="cancelled"/>
4585                                                     <enumeration value="cancel-contradiction"/>
4586                                                     <enumeration value="confirm-contradiction"/>
4587                                                     <enumeration value="hazard"/>
4588                                                     <enumeration value="invalid"/>
4589                                                 </restriction>
4590                                             </simpleType>
4591                                         </element>
4592                                         <element ref="btp:qualifiers" minOccurs="0"/>
4593                                     </sequence>
4594                                 </complexType>
4595                             </element>
4596                         </sequence>
4597                     </complexType>
4598                 </element>
4599                 <element ref="btp:qualifiers" minOccurs="0"/>
4600             </sequence>
4601             <attribute name="id" type="ID"/>
4602         </complexType>
4603     </element>
4604
4605
4606 </schema>
4607

```

XML schema for standard qualifiers

4608
4609
4610
4611
4612
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

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

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

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

```
4660         <element name="minimum-timeout"  
4661 type="nonNegativeInteger"/>  
4662         </sequence>  
4663     </extension>  
4664 </complexContent>  
4665 </complexType>  
4666 </element>  
4667  
4668 <element name="inferior-name" substitutionGroup="btp:qualifier">  
4669     <complexType>  
4670         <complexContent>  
4671             <extension base="btp:qualifier-type">  
4672                 <sequence>  
4673                     <element name="inferior-name" type="string"/>  
4674                 </sequence>  
4675             </extension>  
4676         </complexContent>  
4677     </complexType>  
4678 </element>  
4679  
4680 </schema>  
4681
```

4681
4682

4683 **Carrier Protocol Bindings**

4684
4685
4686
4687
4688
4689
4690
4691
4692
4693
4694

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

4695 **Carrier Protocol Binding Proforma**

4696
4697
4698

A BTP carrier binding specification should provide the following information:

4699 **Binding name:** A name for the binding, as used in the “binding name” field of BTP
4700 addresses (and available for declaring the capabilities of an implementation). Binding
4701 specified in this document, and future revisions of this document have binding names that are
4702 simple strings of letters, numbers and hyphens (and, in particular, do not contain colons).
4703 Bindings specified elsewhere shall have binding names that are URIs. Bindings specified in
4704 this document use numbers to identify the version of the binding, not the version(s) of the
4705 carrier protocol.

4706
4707
4708
4709

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

4710
4711
4712
4713

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

4714
4715
4716
4717

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

4722
4723
4724
4725
4726

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

4727 “the CONTEXT may be carried as a parameter of an application invocation”). Alternatively,
4728 the binding may specify a general method that represents the relationship between application
4729 and BTP messages.

4730

4731 **Implicit messages:** This section specifies which BTP messages, if any, are not sent explicitly
4732 but are treated as implicit in [carrier-protocol mechanisms](#), application messages or other BTP
4733 messages. This may depend on particular parameter values of the BTP messages or the
4734 application messages.

4735

4736 **Faults:** The relationship between the fault and exception reporting mechanisms of the carrier
4737 protocol and of BTP shall be defined. This may include definition of which carrier protocol
4738 exceptions are equivalent to a FAULT/communication-failure message.

4739

4740 **Relationship to other bindings:** Any relationship to other bindings is defined in this section.
4741 If BTP addresses with different bindings are be considered to match (for purposes of
4742 identifying the peer Superior/Inferior and redirection), this should be specified here.

4743

4744 **Limitations on BTP use:** Any limitations on the full range of BTP functionality that are
4745 imposed by use of this binding should be listed. This would include limitations on which
4746 messages can be sent, which event sequences are supported and restrictions on parameter
4747 values. Such limitations may reduce the usefulness of an implementation, but may be
4748 appropriate in certain environments.

4749

4750 **Other:** Other features of the binding, especially any that will potentially affect interoperation
4751 should be specified here. This may include restrictions or requirements on the use or support
4752 of optional carrier parameters or mechanisms.

4753

4754 **Bindings for request/response carrier protocols**

4755

4756 BTP does not generally follow a request/response pattern. In particular, on the outcome
4757 relationship either side may initiate a message – this is an essential part of the presume-abort
4758 recovery paradigm although it is not limited to recovery cases. However, there are some BTP
4759 messages, especially in the control relationship, that do have a request/response pattern.
4760 Many (potential) carrier protocols (e.g. HTTP) do have a request/response pattern. The
4761 specification of a binding specification to a request/response carrier protocol needs to state
4762 what rules apply – which messages can be carried by requests, which by responses. The
4763 simplest rule is to send all BTP messages on requests, and let the carrier responses travel back
4764 empty. This would be inefficient in use of network resources, and possibly inconvenient
4765 when used for the BTP request/response pairs.

4766

4767 This section defines a set of rules that allow more efficient use of the carrier, while allowing
4768 the initiator of a BTP request/response pair to ensure the BTP response is sent back on the
4769 carrier response. These rules are specified in this section to enable binding specifications to
4770 reference them, without requiring each binding specification to repeat similar information.

4771

4772 [These rules also allow the receiver of a message between Superior and Inferior \(in either direction\) on a carrier protocol request to send any reply message on the carrier response –](#)

4773 [the “sender-address” field is implicitly considered to be that of the sender of the carrier](#)
4774 [request.](#)

4775
4776 A binding to a request/response carrier is not required to use these rules. It may define other
4777 rules.

4778 4779 Request/response exploitation rules

4780
4781 These rules allow implementations to use the request and response of the carrier protocol
4782 efficiently, and, when a BTP request/response exchange occurs, to either treat the
4783 request/response exchanges of the carrier protocol and of BTP independently, if both sides
4784 wish, or allow either side to map them closely.

4785
4786 Under these rules, an implementation sending a BTP request (i.e. a message, other than
4787 CONTEXT, which has “reply-address” as a parameter in the abstract message definition), can
4788 ensure that it and the reply map to a carrier request/response by supplying no value for the
4789 “reply-address”. An implementation receiving such a request is required to send the BTP
4790 response on the carrier response.

4791
4792 Conversely, if an implementation does supply a “reply-address” value on the request, the
4793 receiver has the option of sending the BTP response back on the carrier response, or sending
4794 it on a new carrier request.

4795
4796 Within the outcome relationship, apart from ENROL/~~ENROLLED~~, there is no “reply-
4797 address”, and the parties normally know each other’s “~~superior-address-as-superior~~” and
4798 “~~inferior-address-as-inferior~~”. [However, these messages have a “sender-address”, which is](#)
4799 [used when the receiver does not have knowledge of the peer. In this case, the “sender-](#)
4800 [address” is treated as the “reply-address” of the other messages – if the field is absent in a](#)
4801 [message on a carrier request, the “sender-address” is implicitly that of the request sender.](#)
4802 [Any message for the peer \(including the three messages mentioned, FAULT but also any](#)
4803 [other valid message in the Superior:Inferior relationship\) may be sent on the carrier response.](#)
4804 [Apart from this, b](#)Both sides are permitted to treat the carrier request/response exchanges as
4805 ~~just~~ opportunities for sending messages to the appropriate destination.

4806
4807 The rules:

- 4808
- 4809 a) A BTP actor **may** bundle one or more BTP messages and related groups that
4810 have the same binding address for their target in a single btp:messages and
4811 transmit this btp:messages element on a carrier protocol request. There is no
4812 restriction on which combinations of messages and groups may be so bundled,
4813 other than that they have the same binding address, and that this binding address
4814 is usable as the destination of a carrier protocol request.
 - 4815
4816 b) A BTP actor that has received a carrier protocol request to which it has not yet
4817 responded, and which has one or more BTP messages and groups whose binding
4818 address for the target matches the origin of the carrier request **may** bundle such

- 4819 BTP messages in a single btp:messages element and transmit that on the carrier
4820 protocol response.
- 4821
- 4822 c) A BTP actor that has received, on a carrier protocol request, one or more BTP
4823 messages or related groups that require a BTP response and for which no “reply-
4824 address” was supplied, **must** bundle the responding BTP message and groups in a
4825 btp:messages element and transmit this element on the carrier protocol response
4826 to the request that carried the BTP request.
- 4827
- 4828 d) A BTP actor that has received, on a carrier protocol request, one or more BTP
4829 messages or related groups that, as abstract messages, have a “sender-address”
4830 parameter but no “reply-address” was supplied and does not have knowledge of
4831 the peer address, **must** bundle the responding BTP message and groups in a
4832 btp:messages element and transmit this element on the carrier protocol response
4833 to the request that carried the BTP request. If the actor does have knowledge of
4834 the peer address it **may** send one or messages for the peer in the carrier protocol
4835 response, regardless of whether the binding address of the peer matches the
4836 address of the carrier protocol requestor.
- 4837
- 4838 ~~e)~~ Where only one message or group is to be sent, it shall be contained within a
4839 btp:messages element, as a bundle of one element.
- 4840
- 4841 ~~e)f)~~ A BTP actor that receives a carrier protocol request carrying BTP messages that
4842 do have a “reply-address”, or which initiate processing that produces BTP
4843 messages whose target binding address matches the origin of the request, **may**
4844 freely choose whether to use the carrier protocol response for the replies, or to
4845 send back an “empty carrier protocol response”, and send the BTP replies in a
4846 separately initiated carrier protocol request. The characteristics of an “empty
4847 carrier protocol response” shall be stated in the particular binding specification.
- 4848
- 4849 g) A BTP actor that sends BTP messages on a carrier protocol request **must** be able
4850 to accept returning BTP messages on the corresponding carrier protocol response
4851 and, if the actor has offered an address on which it will receive carrier requests,
4852 must be able to accept “replying” BTP messages on a separate carrier protocol
4853 request.
- 4854

4855 SOAP Binding

4856

4857 This binding describes how BTP messages will be carried using SOAP as in the [SOAP 1.1](#)
4858 specification, using the SOAP literal messaging style conventions. If no application message
4859 is sent at the same time, the BTP messages are contained within the SOAP Body element. If
4860 application messages are sent, the BTP messages are contained in the SOAP Header element.

4861

4862 **Binding name:** soap-http-1

4863

4864 **Binding address format:** shall be a URL, of type HTTP.

4865

4866 **BTP message representation:** The string representation of the XML, as specified in the
4867 XML schema defined in this document shall be used. The BTP XML messages are embedded
4868 in the SOAP message without the use of any specific encoding rules (literal style SOAP
4869 message); hence the encodingStyle attribute need not be set or can be set to an empty string.
4870

4871 **Mapping for BTP messages (unrelated):** The “request/response exploitation” rules shall be
4872 used.
4873

4874 BTP messages sent on an HTTP request or HTTP response which is not carrying an
4875 application message, the messages are contained in a single btp:messages element which is
4876 the immediate child element of the SOAP Body element.
4877

4878 An “empty carrier protocol response” sent after receiving an HTTP request containing a
4879 btp:messages element in the SOAP Body and the implementation BTP actor chooses just to
4880 reply at the lower level (and when the request/response exploitation rules allow an empty
4881 carrier protocol response), shall be any of:

- 4882 a) an empty HTTP response
- 4883 b) an HTTP response containing an empty SOAP Envelope
- 4884 c) an HTTP response containing a SOAP Envelope containing a single, empty
4885 btp:messages element.
4886

4887 The receiver (the initial sender of the HTTP request) shall treat these in the same way – they
4888 have no effect on the BTP sequence (other than indicating that the earlier sending did not
4889 cause a communication failure.)
4890

4891

4892

4893 If an application message is being sent at the same time, the mapping for related messages
4894 shall be used, as if the BTP messages were related to the application message. (There is no
4895 ambiguity in whether the BTP messages are related, because only CONTEXT and ENROL
4896 can be related to an application message.)
4897

4898 **Mapping for BTP messages related to application messages:** All BTP messages sent with
4899 an application message, whether related to the application message or not, shall be sent in a
4900 single btp:messages element in the SOAP Header. There shall be precisely one btp:messages
4901 element in the SOAP Header.
4902

4903 The “request/response exploitation” rules shall apply to the BTP messages carried in the
4904 SOAP Header, as if they had been carried in a SOAP Body, unrelated to an application
4905 message, sent to the same binding address.

4906 Note – The application protocol itself (which is using the SOAP Body) may
4907 use the SOAP RPC or document approach – this is determined by the
4908 application.

4909 Only CONTEXT and ENROL messages are related (&) to application messages. If there is
4910 only one CONTEXT or one ENROL message present in the SOAP Header, it is assumed to

4911 be related to the whole of the application message in the SOAP Body. If there are multiple
4912 CONTEXT or ENROL messages, any relation of these BTP messages shall be indicated by
4913 application specific means.

4914 Note 1 – An application protocol could use references to the ID values of the
4915 BTP messages to indicate relation between BTP CONTEXT or ENROL
4916 messages and the application message.

4917 Note 2 -- However indicated, what the relatedness means, or even whether it
4918 has any significance at all, is a matter for the application.

4919
4920 **Implicit messages:** A SOAP FAULT, or other communication failure received in response to
4921 a SOAP request that had a CONTEXT in the SOAP Header shall be treated as if a
4922 CONTEXT_REPLY/repudiated had been received. See also the discussion under “other”
4923 about the SOAP mustUnderstand attribute.
4924

4925 **Faults:** A SOAP FAULT or other communication failure shall be treated as
4926 FAULT/communication-failure.
4927

4928 **Relationship to other bindings:** A BTP address for Superior or Inferior that has the binding
4929 string “soap-http-1” is considered to match one that has the binding string “soap-attachments-
4930 http-1” if the binding address and additional information fields match.
4931

4932 **Limitations on BTP use:** None
4933

4934 **Other:** The SOAP BTP binding does not make use of SOAPAction HTTP header or actor
4935 attribute. The SOAPAction HTTP header is left to be application specific when there are
4936 application messages in the SOAP Body, as an already existing web service that is being
4937 upgraded to use BTP might have already made use of SOAPAction. The SOAPAction HTTP
4938 header shall be omitted when the SOAP message carries only BTP messages in the SOAP
4939 Body.
4940

4941 The SOAP mustUnderstand attribute, when used on the btp:messages containing a BTP
4942 CONTEXT, ensures that the receiver (server, as a whole) supports BTP sufficiently to
4943 determine whether any enrolments are necessary and replies with CONTEXT_REPLY as
4944 appropriate. The sender of the CONTEXT (and related application message) can use this to
4945 ensure that the application work is performed as part of the business transaction, assuming the
4946 receiver’s SOAP implementation supports the mustUnderstand attribute. If mustUnderstand if
4947 false, a receiver can ignore the CONTEXT (if BTP is not supported there), and no
4948 CONTEXT_REPLY will be returned. It is a local option on the sender (client) side whether
4949 the absence of a CONTEXT_REPLY is assumed to be equivalent to aCONTEXT_REPLY/ok
4950 (and the business transaction allowed to proceed to confirmation).
4951

4952 Note – some SOAP implementations may not support the mustUnderstand attribute sufficiently to
4953 enforce these requirements.

4954
4955
4956
4957
4958
4959
4960
4961
4962
4963
4964
4965
4966
4967
4968
4969
4970
4971
4972
4973
4974
4975
4976
4977
4978
4979
4980
4981
4982
4983
4984
4985
4986
4987
4988
4989
4990
4991
4992
4993
4994
4995
4996
4997
4998
4999
5000
5001
5002
5003
5004
5005

Example scenario using SOAP binding

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

```
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  soap:encodingStyle="">
  <soap:Header>
    <btp:messages xmlns:btp="urn:oasis:names:tc:BTP:1.0:core" >
      <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:additional-information>btpengine</btp:additional-
information>
          </btp:superior-address>
          <btp:superior-
identifier>http://example.com/1001</btp:superior-identifier>
          <btp:qualifiers>
            <btpq:transaction-timelimit
xmlns:btpq="urn:oasis:names:tc:BTP:1.0:qualifiers"><btpq:timelimit
>1800</btpq:timelimit></btpq:transaction-timelimit>
            </btp:qualifiers>
          </btp:context>
        </btp:messages>
      </soap:Header>
      <soap:Body>
        <ns1:orderGoods
xmlns:ns1="http://example.com/2001/Services/xyzgoods">
          <custID>ABC8329045</custID>
          <itemID>224352</itemID>
          <quantity>5</quantity>
        </ns1:orderGoods>
      </soap:Body>
    </soap:Envelope>
```

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

5006 CONTEXT_REPLY & ENROL omit the “target-address” (the receiver of this example
5007 remembers the superior address from the original CONTEXT)

```
5008
5009 <soap:Envelope
5010     xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
5011     soap:encodingStyle="">
5012
5013     <soap:Header>
5014     </soap:Header>
5015
5016     <soap:Body>
5017
5018         <btp:messages xmlns:btp="urn:oasis:names:tc:BTP:1.0:corexml">|
5019             <btp:related-group>
5020                 <btp:context-reply>
5021                     <btp:target-additional-information>btpengine</btp:target-
5022 additional-information>
5023                 <btp:superior-
5024 identifier>http://example.com/1001</btp:superior-identifier>
5025                 <completion-status>related</completion-status>
5026                 </btp:context-reply>
5027
5028                 <btp:enrol response-requested="false">
5029                     <btp:target-additional-
5030 information>btpengine</btp:target-additional-information>
5031                     <btp:superior-
5032 identifier>http://example.com/1001</btp:superior-identifier>
5033                     <btp:inferior-address>
5034                         <btp:binding>soap-http-1</btp:binding>
5035                         <btp:binding-address>
5036                             http://services.example.com/soaphandler
5037                         </btp:binding-address>
5038                     </btp:inferior-address>
5039                     <btp:inferior-identifier>
5040                         http://example.com/AAAB
5041                     </btp:inferior-identifier>
5042                 </btp:enrol>
5043
5044                 </btp:related-group>
5045
5046             </btp:messages>
5047
5048         </soap:Body>
5049
5050 </soap:Envelope>
```

5051
5052

5053 SOAP + Attachments Binding

5054
5055 This binding describes how BTP messages will be carried using SOAP as in the [SOAP](#)
5056 [Messages with Attachments](#) specification. It is a superset of the Basic SOAP binding, soap-
5057 http-1. The two bindings only differ when application messages are sent.

5058
5059
5060
5061
5062
5063
5064
5065
5066
5067
5068
5069
5070
5071
5072
5073
5074
5075
5076
5077
5078
5079
5080
5081
5082
5083
5084
5085
5086
5087
5088
5089
5090
5091
5092
5093
5094
5095
5096
5097
5098
5099
5100
5101
5102
5103
5104
5105

Binding name: soap-attachments-http-1

Binding address format: as for soap-http-1

BTP message representation: As for soap-http-1

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

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

Implicit messages: As for soap-http-1.

Faults: As for soap-http-1.

Relationship to other bindings: A BTP address for Superior or Inferior that has the binding string “soap-http-1” is considered to match one that has the binding string “soap-attachments-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=" " >
```

```

5106     <soap:Header>
5107
5108         <btp:messages xmlns:btp="urn:oasis:names:tc:BTP:1.0:corexml">|
5109             <btp:context superior-type="atom">
5110                 <btp:superior-address>
5111                     <btp:binding>soap-http-1</btp:binding>
5112                     <btp:binding-address>
5113                         http://client.example.com/soaphandler
5114                     </btp:binding-address>
5115                 </btp:superior-address>
5116                 <btp:superior-
5117 identifier>http://example.com/1001</btp:superior-identifier>
5118             </btp:context>
5119         </btp:messages>
5120
5121     </soap:Header>
5122
5123     <soap:Body>
5124         <orderGoods href="cid:anotherID"/>
5125     </soap:Body>
5126
5127 </soap:Envelope>
5128
5129 --MIME_boundary
5130 Content-Type: text/xml
5131 Content-ID: anotherID
5132
5133     <ns1:orderGoods
5134 xmlns:ns1="http://example.com/2001/Services/xyzgoods">
5135         <custID>ABC8329045</custID>
5136         <itemID>224352</itemID>
5137         <quantity>5</quantity>
5138     </ns1:orderGoods>
5139
5140
5141 --MIME_boundary--
5142
5143

```

5144 Conformance

5145 A BTP implementation need not implement all aspects of the protocol to be useful. The level
5146 of conformance of an implementation is defined by which roles it can support using the
5147 specified messages and carrier protocol bindings for interoperation with other
5148 implementations.
5149

5150
5151 A partially conformant implementation may implement some roles in a non-interoperable
5152 way, giving that implementation's users comparable proprietary functionality.
5153

5154 The following Roles and Role Groups are used to define conformance:
5155

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

5156
5157
5158
5159
5160

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	Cohesive Superior Participant
Atomic Coordination Hub	Initiator/Terminator Atomic Coordination Hub Participant
Cohesive Coordination Hub	Initiator/Terminator Cohesive Coordination Hub Participant

5161
5162
5163
5164
5165
5166
5167
5168
5169
5170

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

5170 Part 3. Appendices

5171

5172 The glossary is the subject of issue 4

5173

5174 **A. Glossary**

5175

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 Inferior-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 superior-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.
<u>superior</u>-address-as-superior	The address used to communicate with an actor playing the role of an Superior
<u>Composer</u>-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.
<u>Inferior</u>-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 <u>superior</u> -address- as-superior of a given Superior.
Identity-as-Inferior	The combination of inferior-identifier and <u>inferior</u> -address- as-inferior of a given Inferior.

5176