

Business Transaction Protocol

An OASIS Committee Specification

CURRENT STATUS : internal committee draft

Version 1.0 [0.9.2.1]

DD Mmm 2002 [15 February 2002 23:26]

<i>Working draft 0.1 (pre-London)</i>	14 June 2001
<i>Working draft 0.2 (London)</i>	18 June 2001
<i>Working draft 0.3a (circulated)</i>	12 July 2001
<i>Working draft 0.3c (circulated)</i>	20 July 2001
<i>Working draft 0.4 (circulated; incorporates PRF material)</i>	25 July 2001
<i>Working draft 0.6 (State tables)</i>	31 August 2001
<i>Working Draft 0.9</i>	24 October 2001
<i>Working Draft 0.9.0.1 – minor editorials issues applied</i>	16 November 2001
<i>Working Draft 0.9.0.2 – issue resolutions balloting to 10 Dec 2001</i>	4 December 2001
<i>Working Draft 0.9.0.3 – possible solution to msging issues</i>	11 December 2001
<i>Working Draft 0.9.0.4 – issue 79 solution, revise msging issues</i>	12 January 2002
<i>Working Draft 0.9.1 – includes all issues agreed 16 Jan 2002, and 82 (deferred)</i>	18 January 2002
<i>Working Draft 0.9.1.1 – format changes and proposed soln 77,78, 17.</i>	27 January 2002
<i>Working Draft 0.9.1.2 – xml changes, new schema, and issue 74</i>	30 January 2002
<i>Working Draft 0.9.1.3 – corrections, issue 30, state table – 81, 104</i>	8 February 2002
<i>Working Draft 0.9.2 – all issues as agreed 13 February 2002</i>	13 February 2002
<i>Working Draft 0.9.2.1 – issue 2, 3, 15, 19, 50, 67, 95</i>	15 February 2002

[Change marks relative to 0.9.2](#)

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

```
<ntp:begin> ... </ntp:begin>
```

136

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

138

139

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

140

```
{
```

141

```
}
```

142

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

146
147
148
149
150

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

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

196	ENROLLED	39
197	RESIGN	40
198	RESIGNED	41
199	PREPARE	42
200	PREPARED	42
201	CONFIRM	44
202	CONFIRMED	44
203	CANCEL	45
204	CANCELLED	46
205	CONFIRM_ONE_PHASE	47
206	HAZARD	48
207	CONTRADICTION	49
208	SUPERIOR_STATE	50
209	INFERIOR_STATE	51
210	REDIRECT	53
211	Messages used in control relationships	54
212	BEGIN	54
213	BEGUN	55
214	PREPARE_INFERIORS	56
215	CONFIRM_TRANSACTION	58
216	TRANSACTION_CONFIRMED	60
217	CANCEL_TRANSACTION	60
218	CANCEL_INFERIORS	61
219	TRANSACTION_CANCELLED	62
220	REQUEST_INFERIOR_STATUSES	63
221	INFERIOR_STATUSES	64
222	Groups – combinations of related messages	66
223	CONTEXT & application message	66
224	CONTEXT_REPLY & ENROL	67
225	CONTEXT_REPLY (& ENROL) & PREPARED / & CANCELLED	68
226	CONTEXT_REPLY & ENROL & application message (& PREPARED)	69
227	BEGUN & CONTEXT	70
228	BEGIN & CONTEXT	70
229	Standard qualifiers	70
230	Transaction timelimit	70
231	Inferior timeout	71
232	Minimum inferior timeout	72
233	Inferior name	72
234	State Tables	74
235	Explanation of the state tables	74
236	Status queries	74
237	Decision events	74
238	Disruptions – failure events	75
239	Invalid cells and assumptions of the communication mechanism	75
240	Meaning of state table events	76
241	Persistent information	80
242	Failure Recovery	93

243	Types of failure	93
244	Persistent information	94
245	Redirection.....	95
246	Terminator:Decider failures.....	96
247	XML representation of Message Set.....	96
248	Addresses	97
249	Qualifiers	97
250	Identifiers	98
251	Message References.....	98
252	Messages.....	98
253	CONTEXT.....	98
254	CONTEXT_REPLY	98
255	REQUEST_STATUS	99
256	STATUS	99
257	FAULT.....	99
258	ENROL	100
259	ENROLLED	101
260	RESIGN	101
261	RESIGNED.....	101
262	PREPARE	102
263	PREPARED	102
264	CONFIRM	102
265	CONFIRMED	102
266	CANCEL	103
267	CANCELLED.....	103
268	CONFIRM_ONE_PHASE	103
269	HAZARD.....	104
270	CONTRADICTION.....	104
271	SUPERIOR_STATE.....	104
272	INFERIOR_STATE.....	104
273	REDIRECT.....	105
274	BEGIN	105
275	BEGUN.....	106
276	PREPARE_INFERIORS	106
277	CONFIRM_TRANSACTION	106
278	TRANSACTION_CONFIRMED.....	107
279	CANCEL_TRANSACTION	107
280	CANCEL_INFERIORS	107
281	TRANSACTION_CANCELLED.....	108
282	REQUEST_INFERIOR_STATUSES	108
283	INFERIOR_STATUSES	108
284	Standard qualifiers	109
285	Transaction timelimit.....	109
286	Inferior timeout	109
287	Minimum inferior timeout	109
288	Inferior name.....	109
289	Compounding of Messages.....	110

290	XML Schemas	111
291	XML schema for BTP messages.....	111
292	XML schema for standard qualifiers	124
293	Carrier Protocol Bindings	126
294	Carrier Protocol Binding Proforma.....	126
295	Bindings for request/response carrier protocols	127
296	Request/response exploitation rules.....	128
297	SOAP Binding	129
298	Example scenario using SOAP binding	131
299	SOAP + Attachments Binding.....	133
300	Conformance	135
301	Part 3. Appendices.....	138
302	A. Glossary.....	138
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 CONFIRM_COMPLETE
808 CANCEL_COMPLETE
809 INFERIOR_STATUSES

810
811

812 Coordinator

813

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

816

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

818

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

823

824 Composer

825

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

828

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

831

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

836

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

841

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

844

845

846 Terminator

847

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

850

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

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

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

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

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

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

880
881 A Terminator receives
882 CONFIRM_COMPLETE
883 CANCEL_COMPLETE
884 INFERIOR_STATUSES

885 886 Initiator

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

891
892 An Initiator sends
893
894 BEGIN
895 BEGIN & CONTEXT
896
897 to a Factory, and receives in reply
898
899 BEGUN & CONTEXT

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

Factory

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

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

A Factory receives

BEGIN
BEGIN & CONTEXT

and replies with

BEGUN & CONTEXT

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

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

Other roles

Redirector

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

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

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

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

948

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

951

952 **Status Requestor**

953

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

959

960 A Status Requestor sends

961

962 REQUEST_STATUS

963 REQUEST_INFERIOR_STATUSES

964

965 and receives

966

967 STATUS

968 INFERIOR_STATUSES

969

970 in response.

971

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

975

976 **Abstract Messages and Associated Contracts**

977

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

981

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

983

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

986

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

989

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

991

992 and

993

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

995

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

998

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

1005

1006

1007 **Addresses**

1008

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

1013

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

1022

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

1035

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

1037

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

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

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

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

1061 **Request/response pairs**

1062

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

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

1073 **Compounding messages**

1074

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

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

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

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

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

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

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

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

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

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

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

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

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

1158

1159 **Extensibility**

1160

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

1170

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

1173

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

1175

1176 **Messages**

1177

1178 **Qualifiers**

1179

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

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

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

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

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

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

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

1210
1211
1212 **Messages not restricted to outcome or control relationships.**
1213

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

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

1220

1221 CONTEXT

1222

1223 A CONTEXT is supplied by (or on behalf of) a Superior and related to one or more
1224 application messages. (The means by which this relationship is represented is determined by
1225 the binding and the binding mechanisms of the application protocol.) The “superior_type”
1226 parameter identifies whether the Superior will apply the same decision to all Inferiors
1227 enrolled using the same superior identifier (“superior_type” is “atom”) or whether it may
1228 apply different decisions (“superior_type” is “cohesion”).
1229

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

1230

1231

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

1234

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

1240

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

1243

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

1246

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

1249

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

1252

1253

1254 CONTEXT_REPLY

1255

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

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

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

1268 **superior-identifier** the “superior-identifier” from the CONTEXT
 1269

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

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

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

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

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

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

1287
1288
1289
1290
1291
1292
1293

REQUEST_STATUS

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

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

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

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

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

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

qualifiers standardised or other qualifiers.

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

1309
1310
1311
1312
1313
1314
1315
1316

General

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

UnknownTransaction – if the target-identifier is unknown

STATUS

1317
1318
1319
1320
1321

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

Parameter	Type
target-address	BTP address
responders-identifier	Identifier

status See below
 qualifiers List of qualifiers

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

target_address the address to which the STATUS is sent. This will be the “reply_address” on the REQUEST_STATUS message

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

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

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

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

1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346

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
qualifiers	List of qualifiers

1347
1348 **target_address** the address to which the FAULT is sent. This may be the
1349 “reply_address” from a received message or the address of the opposite side
1350 (superior/inferior) as given in a CONTEXT or ENROL message
1351
1352 **superior_identifier** the “superior_identifier” as on the CONTEXT message and
1353 as used on the ENROL message (present only if the FAULT is sent to the
1354 superior).
1355
1356 **inferior_identifier** the “inferior_identifier” as on the ENROL message (present
1357 only if the FAULT is sent to the inferior)
1358
1359 **fault_type** identifies the nature of the error, as specified for each of the main
1360 messages.
1361
1362 **fault_data** information relevant to the particular error. Each “fault_type” defines
1363 the content of the “fault_data”.
1364

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

1366
 1367 *UnknownParameter* A BTP message has been received with an unrecognised parameter Free text explanation
 1368
 1369 **q**
 1370 **u**
 1371 **Qualifiers** standardised or other qualifiers.
 1372

1373 Note – If the carrier mechanism used for the transmission of BTP messages
 1374 is capable of delivering messages in a different order than they were sent in,
 1375 the “WrongState” FAULT is not sent and should be ignored if received.

1376
 1377 **REQUEST_INFERIOR_STATUSES, INFERIOR_STATUSES**
 1378

1379 REQUEST_INFERIOR_STATUSES may be sent to and INFERIOR_STATUSES sent from
 1380 any Decider, Superior or Inferior, asking it to report on the status of its relationships with
 1381 Inferiors (if any). Since Deciders are required to respond to
 1382 REQUEST_INFERIOR_STATUSES with INFERIOR_STATUSES but non-Deciders may
 1383 just issue FAULT(StatusRefused), and INFERIOR_STATUSES is also used as a reply to
 1384 other messages from Terminator to Decider, these messages are described below under the
 1385 messages used in the control relationships.
 1386

1387 **Messages used in the outcome relationships**
 1388

1389 **ENROL**
 1390

1391 A request to a Superior to ENROL an Inferior. This is typically issued after receipt of a
 1392 CONTEXT message in relation to an application request.
 1393 The actor issuing ENROL plays the role of Enroller.
 1394

Parameter	type
target_address	BTP address
superior_identifier	Identifier
reply_response-requested	Boolean
reply_address	BTP address
address-as-inferior	Set of BTP addresses
inferior_identifier	Identifier
qualifiers	List of qualifiers

1395
 1396 **target_address** the address to which the ENROL is sent. This will be the
 1397 address-as-superior from the CONTEXT message.

1398
 1399 **superior_identifier**. The “superior_identifier” as on the CONTEXT message
 1400
 1401 **reply_response-requested** true if an ENROLLED response is required, false
 1402 otherwise. Default is false.
 1403
 1404 **reply_address** the address to which a replying ENROLLED is to be sent, if
 1405 “reply_response-requested” is true. If this field is absent and “reply_response-
 1406 requested” is true, the ENROLLED should be sent to the “address-as-inferior”
 1407 (or one of them, at sender’s option)
 1408
 1409 **address-as-inferior** the address to which PREPARE, CONFIRM, CANCEL and
 1410 SUPERIOR_STATE messages for this Inferior are to be sent.
 1411
 1412 **inferior_identifier** an identifier that identifies this Inferior. This shall be globally
 1413 unambiguous..
 1414
 1415 **qualifiers** standardised or other qualifiers. The standard qualifier “Inferior
 1416 name” may be present.
 1417

1418 Types of FAULT possible (sent to “reply_address”)

1419
 1420 *General*

1421 *InvalidSuperior* – if “superior_identifier” is unknown

1422 *DuplicateInferior* – if inferior with at least one of the set address-as-
 1423 inferior the same and the same “inferior_identifier” is already enrolled

1424 *WrongState* – if it is too late to enrol new Inferiors (generally if the
 1425 Superior has already sent a PREPARED message to its superior or
 1426 terminator, or if it has already issued CONFIRM to other Inferiors).
 1427

1428 The form ENROL/rsp-req refers to an ENROL message with “reply_response-requested”
 1429 having the value “true”; ENROL/no-rsp-req refers to an ENROL message with “reply
 1430 response-requested” having the value “false”
 1431

1432 ENROL/no-rsp-req is typically sent in relation to CONTEXT_REPLY/related. ENROL/rsp-
 1433 req is typically when CONTEXT_REPLY/completed will be used (after the ENROLLED
 1434 message has been received.)
 1435

1436 **ENROLLED**

1437
 1438 Sent from Superior in reply to an ENROL/rsp-req message, to indicate the Inferior has been
 1439 successfully enrolled (and will therefore be included in the termination exchanges)
 1440

Parameter	Type
target_address	BTP address

Parameter	Type
inferior_ <u>i</u> dentifier	Identifier
Qualifiers	List of qualifiers

1441

1442

1443

1444

1445

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

1446

inferior_identifier The "inferior_identifier" as on the ENROL message

1447

1448

qualifiers standardised or other qualifiers.

1449

No FAULT messages are issued on receiving ENROLLED.

1451

1452

1453 RESIGN

1454

1455

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

1456

1457

1458

1459

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

1460

1461

1462

Parameter	type
target_ <u>a</u> ddress	BTP address
superior_ <u>i</u> dentifier	identifier
inferior_ <u>i</u> dentifier	identifier
response_ <u>r</u> equested	Boolean
Qualifiers	List of qualifiers

1463

1464

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

1465

1466

superior-identifier The "inferior_identifier" as on the ENROL message

1467

1468

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

1469

1470

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

1471

1472

1473

1474 **qualifiers** standardised or other qualifiers.

1475

1476 Note -- RESIGN is equivalent to readonly vote in some other protocols, but can be issued
1477 early.

1478

1479 Types of FAULT possible (sent to address-as-inferior)

1480

1481

General

1482

InvalidSuperior – if “superior_identifier” is unknown

1483

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

1484

1485

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

1486

1487

1488 The form RESIGN/rsp-req refers to an RESIGN message with “reply_response-requested”
1489 having the value “true”; RESIGN /no-rsp-req refers to an RESIGN message with “reply
1490 response-requested” having the value “false”

1491

1492

1493 **RESIGNED**

1494

1495 Sent in reply to a RESIGN/rsp-req message.

1496

Parameter	Type
target_ <u>address</u>	BTP address
inferior_ <u>identifier</u>	Identifier
qualifiers	List of qualifiers

1497

1498

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

1499

1500

1501

inferior_identifier The “inferior_identifier” as on the earlier ENROL message
for this Inferior.

1502

1503

1504

qualifiers standardised or other qualifiers.

1505

1506

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

1507

1508

1509

Types of FAULT possible (sent to Superior address)

1510

General

1511

WrongState - if RESIGN has not been sent

1512

No FAULT messages are issued on receiving RESIGNED.

1513

1514 **PREPARE**

1515
1516
1517
1518
1519
1520

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

Parameter	Type
target_address	BTP address
inferior_identifier	Identifier
qualifiers	List of qualifiers

1521
1522
1523
1524
1525
1526
1527
1528
1529
1530

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

inferior_identifier When sent from Superior to Inferior, the “inferior_identifier” as on the earlier ENROL message.

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

1531
1532
1533

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

1534
1535

Types of FAULT possible (sent to Superior address)

1536
1537
1538
1539
1540
1541
1542
1543
1544

General

InvalidInferior – if “inferior_identifier” is unknown, or an inferior-handle on the inferiors-list is unknown

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

1545 **PREPARED**

1546
1547
1548
1549
1550
1551
1552
1553

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

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

1554

1555 **target-address** the address to which the PREPARED is sent. This will be the
 1556 Superior address as on the ENROL message.

1557

1558 **superior-identifier** the “superior-identifier” as on the ENROL message

1559

1560 **inferior-identifier** The “inferior-identifier” as on the ENROL message

1561

1562 **default-is cancel** if “true”, the Inferior states that if the outcome at the Superior
 1563 is to cancel the operations associated with this Inferior, no further messages need
 1564 be sent to the Inferior. If the Inferior does not receive a CONFIRM message, it
 1565 will cancel the associated operations. The value “true” will invariably be used
 1566 with a qualifier indicating under what circumstances (usually a timeout) an
 1567 autonomous decision to cancel will be made. If “false”, the Inferior will expect
 1568 a CONFIRM or CANCEL message as appropriate, even if qualifiers indicate that
 1569 an autonomous decision will be made.

1570

1571 **qualifiers** standardised or other qualifiers. The standard qualifier “Inferior
 1572 timeout” may be carried by PREPARED.

1573

1574 On sending a PREPARED, the Inferior undertakes to maintain its ability to confirm or cancel
 1575 the effects of the associated operations until it receives a CONFIRM or CANCEL message.
 1576 Qualifiers may define a time limit or other constraints on this promise. The “default-is
 1577 cancel” parameter affects only the subsequent message exchanges and does not of itself state
 1578 that cancellation will occur.

1579

1580 Types of FAULT possible (sent to address-as-inferior)

1581

General

1582

1583 **InvalidSuperior** – if “superior-identifier” is unknown

1584

1585 **InvalidInferior** – if no ENROL has been received for this address-as-
 1586 inferior and identifier, or if RESIGN has been received from this Inferior

1586

1587 The form PREPARED/cancel refers to a PREPARED message with “default-is cancel” =
 1588 “true”. The unqualified form PREPARED refers to a PREPARED message with “default-is
 1589 cancel” = “false”.

1590

1591
1592
1593
1594
1595

CONFIRM

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

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

1596
1597
1598
1599
1600
1601
1602

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

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

qualifiers standardised or other qualifiers.

1603
1604
1605
1606
1607
1608

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

Types of FAULT possible (sent to Superior address)

1609
1610
1611
1612
1613
1614

General

InvalidInferior – if “inferior-identifier” is unknown

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

1615
1616

CONFIRMED

1617
1618
1619
1620
1621
1622
1623

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

Parameter	Type
target-address	BTP address
superior-identifier	Identifier
inferior-identifier	Identifier
confirm-received	Boolean

Parameter	Type
qualifiers	List of qualifiers
1624	
1625	target₋address the address to which the CONFIRMED is sent. This will be the
1626	Superior address as on the CONTEXT message.
1627	
1628	superior₋identifier the “superior ₋ identifier” as on the CONTEXT message.
1629	
1630	inferior₋identifier the “inferior ₋ identifier” as on the earlier ENROL message.
1631	
1632	
1633	confirm₋received “true” if CONFIRMED is sent after receiving a CONFIRM
1634	message; “false” if an autonomous confirm decision has been made and either if
1635	no CONFIRM message has been received or the implementation cannot
1636	determine if CONFIRM has been received (due to loss of state information in a
1637	failure).
1638	
1639	qualifiers standardised or other qualifiers.

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

General

InvalidSuperior – if “superior₋identifier” is unknown

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

1648 Note – A CONFIRMED message arriving before a CONFIRM message is
1649 sent, or after a CANCEL has been sent will occur when the Inferior has
1650 taken an autonomous decision and is not regarded as occurring in the wrong
1651 state. (The latter will cause a CONTRADICTION message to be sent.)

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

CANCEL

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

Parameter	Type
target ₋ address	BTP address

inferior_ <u>identifier</u>	Identifier
qualifiers	List of qualifiers

1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700

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

inferior_identifier the "inferior_identifier" as on the earlier ENROL message.

qualifiers standardised or other qualifiers.

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

Types of FAULT possible (sent to Superior address)

General

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

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

CANCELLED

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

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

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

Parameter

target_ <u>address</u>	BTP address
superior_ <u>identifier</u>	Identifier

inferior_ <u>i</u> dentifier	Identifier
qualifiers	List of qualifiers

1701
1702
1703
1704
1705
1706
1707
1708
1709

target_address the address to which the CANCELLED is sent. This will be the Superior address as on the CONTEXT message.

superior_identifier the “superior_identifier” as on the CONTEXT message.

inferior_identifier ~~W~~the inferior identifier as on the earlier ENROL message.

qualifiers standardised or other qualifiers.

1710

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

1711
1712
1713
1714
1715
1716
1717
1718

General

InvalidSuperior – if “superior_identifier” is unknown

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

WrongState – if CONFIRM has been sent

1719
1720
1721
1722

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

1723
1724
1725

CONFIRM_ONE_PHASE

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

1730

Parameter	Type
target_ <u>a</u> ddress	BTP address
inferior_ <u>i</u> dentifier	Identifier
report-hazard	boolean
qualifiers	List of qualifiers

1731
1732
1733
1734

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

1735 **inferior_identifier** The "inferior_identifier" as on the earlier ENROL message
 1736 for this Inferior.
 1737
 1738 **report hazard** Defines whether the superior wishes to be informed if a mixed
 1739 condition occurs for the operations associated with the Inferior. If "report hazard"
 1740 is "true", the Inferior will reply with HAZARD if a mixed condition occurs, or if
 1741 the Inferior cannot determine that a mixed condition has not occurred. If "report
 1742 hazard" is false, the Inferior will report only its own decision, regardless of
 1743 whether that decision was correctly and consistently applied. Default is false.
 1744

1745 **qualifiers** standardised or other qualifiers.
 1746

1747 CONFIRM_ONE_PHASE can be issued by a Superior to an Inferior from whom
 1748 PREPARED has been received (subject to the requirement that there is only one enrolled
 1749 Inferior).
 1750

1751 Types of FAULT possible (sent to Superior address)
 1752

1753 *General*

1754 *InvalidInferior* – if "inferior_identifier" is unknown

1755 *WrongState* – if a PREPARE has already been sent to this Inferior
 1756

1757 **HAZARD**
 1758

1759 Sent when the Inferior has either discovered a "mixed" condition: that is unable to correctly
 1760 and consistently cancel or confirm the operations in accord with the decision (~~either the~~
 1761 ~~received decision of the superior or its own autonomous decision~~), or when the Inferior is
 1762 unable to determine that a "mixed" condition has not occurred.
 1763

1764 HAZARD is also used to reply to a CONFIRM_ONE_PHASE if the Inferior determines there
 1765 is a mixed condition within its associated operations or is unable to determine that there is not
 1766 a mixed condition.
 1767

1768 Note - If the Inferior makes its own autonomous decision then it signals that
 1769 decision with CONFIRMED or CANCELLED and waits to receive a
 1770 confirmatory CONFIRM or CANCEL, or a CONTRADICTION if the
 1771 autonomous decision by the Inferior was the opposite of that made by the
 1772 Superior.

1773

Parameter	Type
target_ <u>address</u>	BTP address
superior_ <u>identifier</u>	Identifier
inferior_ <u>identifier</u>	Identifier

level	mixed/possible
qualifiers	List of qualifiers

1774

1775

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

1776

1777

1778

superior_identifier The “superior_identifier” as on the ENROL message

1779

1780

1781

inferior_identifier The “inferior_identifier” as on the earlier ENROL message

1782

1783

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.

1784

1785

1786

qualifiers standardised or other qualifiers.

1787

1788

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

1789

1790

General

1791

InvalidSuperior – if “superior_identifier” is unknown

1792

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

1793

1794

1795

1796

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

1797

1798

1799

CONTRADICTION

1800

1801

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.

1802

1803

1804

1805

1806

Parameter	Type
target_address	BTP address
inferior_identifier	Identifier
qualifiers	List of qualifiers

1807

1808

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

1809

1810

1811 **inferior_identifier** The “inferior_identifier” as on the earlier ENROL message
1812 for this Inferior.

1813
1814 **qualifiers** standardised or other qualifiers.

1815
1816 Types of FAULT possible (sent to Superior address)

1817
1818 **General**
1819 **InvalidInferior** – if “inferior_identifier” is unknown
1820 **WrongState** – if neither CONFIRMED or CANCELLED has been sent
1821 by this Inferior

1822 SUPERIOR_STATE

1823
1824 Sent by a Superior as a query to an Inferior when

- 1825
1826
- 1827 1. in the active state
 - 1828
1829 2. there is uncertainty what state the Inferior has reached (due to recovery from
1830 previous failure or other reason).

1831
1832 Also sent by the Superior to the Inferior in response to a received INFERIOR_STATE, in
1833 particular states.

1834

Parameter	Type
target_address	BTP address
inferior_identifier	Identifier
status	<i>see below</i>
reply_response_requested	Boolean
qualifiers	List of qualifiers

1835
1836 **target_address** the address to which the SUPERIOR_STATE message is sent.
1837 This will be the address-as-inferior from the ENROL message.

1838
1839 **inferior_identifier** The “inferior_identifier” as on the earlier ENROL message
1840 for this Inferior.

1841
1842 **status** states the current state of the Superior, in terms of its relation to this
1843 Inferior only.

1844

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)

prepared-received PREPARED has been received from the Inferior, but no outcome is yet available

inaccessible The state information for the Superior, or for its relationship with this Inferior, if it exists, cannot be accessed at the moment. This should be a transient condition

unknown The Inferior is not known – it does not exist from the perspective of the Superior. The Inferior can treat this as an instruction to cancel any associated operations

1845

1846 **Reply-response-requested** true, if SUPERIOR_STATE is sent as a query at the
1847 Superior's initiative; false, if SUPERIOR_STATE is sent in reply to a received
1848 INFERIOR_STATE or other message. Can only be true if status is active or
1849 prepared-received. Default is "false"

1850

1851 **qualifiers** standardised or other qualifiers.

1852

1853 The Inferior, on receiving SUPERIOR_STATE with "**reply-response-requested** = true, should
1854 reply in a timely manner by (depending on its state) repeating the previous message it sent or
1855 by sending INFERIOR_STATE with the appropriate status value.

1856

1857 A status of unknown shall only be sent if it has been determined for certain that the Superior
1858 has no knowledge of the Inferior, or (equivalently) it can be determined that the relationship
1859 with the Inferior was cancelled. If there could be persistent information corresponding to the
1860 Superior, but it is not accessible from the entity receiving an INFERIOR_STATE/*y (or
1861 other) message targeted to the Superior or that entity cannot determine whether any such
1862 persistent information exists or not, the response shall be Inaccessible.

1863

1864 SUPERIOR_STATE/unknown is also used as a response to messages, other than
1865 INFERIOR_STATE/*y that are received when the Inferior is not known (and it is known
1866 there is no state information for it).

1867

1868 The form SUPERIOR_STATE/abcd refers to a SUPERIOR_STATE message status having a
1869 value equivalent to "abcd" (for active, prepared-received, unknown and inaccessible) and
1870 with "**reply-response-requested**" = "false". SUPERIOR_STATE/abcd/y refers to a similar
1871 message, but with "**reply-response-requested**" = "true". The form SUPERIOR_STATE/*y
1872 refers to a SUPERIOR_STATE message with "**reply-response-requested**" = "true" and any
1873 value for status.

1874

1875

1876 INFERIOR_STATE

1877

1878 Sent by an Inferior as a query when in the active state to a Superior, when (due recovery from
1879 previous failure or other reason) there is uncertainty what state the Superior has reached.

1880

1881 Also sent by the Inferior to the Superior in response to a received SUPERIOR_STATE, in
 1882 particular states.
 1883

Parameter	Type
target_address	BTP address
superior_identifier	Identifier
inferior_identifier	Identifier
status	<i>see below</i>
reply_response_requested	Boolean
qualifiers	List of qualifiers

1884
 1885 **target_address** the address to which the INFERIOR_STATE is sent. This will
 1886 be the "target_address" as used the original ENROL message.
 1887

1888 **superior_identifier** The "superior_identifier" as used on the ENROL message
 1889

1890 **inferior_identifier** The "inferior_identifier" as on the ENROL message
 1891

1892 **status** states the current state of the Inferior for the atomic business transaction,
 1893 which corresponds to the last message sent to the Superior by (or in the case of
 1894 ENROL for) the Inferior
 1895

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

1896
 1897 **reply_response_requested** "true" if INFERIOR_STATE is sent as a query at the
 1898 Superior's initiative; "false" if INFERIOR_STATE is sent in reply to a received
 1899 SUPERIOR_STATE or other message. Can only be "true" if "status" is "active"
 1900 or "prepared-received". *Can only be "true" if "status" is "active". Default is*
 1901 *"false"*
 1902

1903 **qualifiers** standardised or other qualifiers.
 1904

1905 The Superior, on receiving INFERIOR_STATE with “reply-response-requested” = “true”,
1906 should reply in a timely manner by (depending on its state) repeating the previous message it
1907 sent or by sending SUPERIOR_STATE with the appropriate status value.

1908
1909 A status of “unknown” shall only be sent if it has been determined for certain that the Inferior
1910 has no knowledge of a relationship with the Superior. If there could be persistent information
1911 corresponding to the Superior, but it is not accessible from the entity receiving an
1912 SUPERIOR_STATE/*y (or other) message targetted on the Inferior or the entity cannot
1913 determine whether any such persistent information exists, the response shall be
1914 “inaccessible”.

1915
1916 INFERIOR_STATE/unknown is also used as a response to messages, other than
1917 SUPERIOR_STATE/*y that are received when the Inferior is not known (and it is known
1918 there is no state information for it).

1919
1920 A SUPERIOR_STATE/INFERIOR_STATE exchange that determines that one or both sides
1921 are in the active state does not require that the Inferior be cancelled (unlike some other two-
1922 phase commit protocols). The relationship between Superior and Inferior, and related
1923 application elements may be continued, with new application messages carrying the same
1924 CONTEXT. Similarly, if the Inferior is prepared but the Superior is active, there is no
1925 required impact on the progression of the relationship between them.

1926
1927 The form INFERIOR_STATE/abcd refers to a INFERIOR_STATE message status having a
1928 value equivalent to “abcd” (for active, unknown and inaccessible) and with “reply-response-
1929 requested” = “false”. INFERIOR_STATE/abcd/y refers to a similar message, but with “reply
1930 response-requested” = “true”. The form INFERIOR_STATE/*y refers to a
1931 INFERIOR_STATE message with “reply-response-requested” = “true” and any value for
1932 status.

1933
1934

1935 REDIRECT

1936
1937 Sent when the address previously given for a Superior or Inferior is no longer valid and the
1938 relevant state information is now accessible with a different address (but the same superior or
1939 “inferior-identifier”).
1940

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

1941

1942 **target-address** the address to which the REDIRECT is sent. This may be the
 1943 “reply-address” from a received message or the address of the opposite side
 1944 (superior/inferior) as given in a CONTEXT or ENROL message
 1945
 1946 **superior-identifier** The “superior-identifier” as on the CONTEXT message and
 1947 used on an ENROL message. (present only if the REDIRECT is sent from the
 1948 Inferior).
 1949
 1950 **inferior-identifier** The “inferior-identifier” as on the ENROL message
 1951
 1952 **old-address** The previous address of the sender of REDIRECT. A match is
 1953 considered to apply if any of the “old-address” values match one that is
 1954 already known.
 1955
 1956 **new-address** The (set of alternatives) “new-address” values to be used for
 1957 messages sent to this entity.
 1958
 1959 **qualifiers** standardised or other qualifiers.
 1960
 1961 If the actor whose address is changed is an Inferior, the “new-address” value
 1962 replaces the address-as-inferior as present in the ENROL.
 1963
 1964 If the actor whose address is changed is a Superior, the “new-address” value
 1965 replaces the Superior address as present in the CONTEXT message (or as present
 1966 in any other mechanism used to establish the Superior:Inferior relationship).
 1967
 1968

1969 **Messages used in control relationships**

1970 **BEGIN**

1971 A request to a Factory to create a new Business Transaction. This may either be a new top-
 1972 level transaction, in which case the Composer or Coordinator will be the Decider, or the new
 1973 Business Transaction may be immediately made the Inferior within an existing Business
 1974 Transaction (thus creating a sub-Composer or sub-Coordinator).
 1975
 1976
 1977

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

1978

1979 **target₋address** the address of the entity to which the BEGIN is sent. How this
1980 address is acquired and the nature of the entity are outside the scope of this
1981 specification.

1982
1983 **reply₋address** the address to which the replying BEGUN and related
1984 CONTEXT message should be sent.

1985
1986 **transaction₋type** identifies whether a new Cohesion or new Atom is to be
1987 created; this value will be the “superior₋type” in the new CONTEXT

1988
1989 **qualifiers** standardised or other qualifiers. The standard qualifier “Transaction
1990 timelimit” may be present on BEGIN, to set the timelimit for the new business
1991 transaction and will be copied to the new CONTEXT. The standard qualifier
1992 “Inferior name” may be present if there is a CONTEXT related to the BEGIN.

1993
1994 A new top-level Business Transaction is created if there is no CONTEXT related to the
1995 BEGIN. A Business Transaction that is to be Inferior in an existing Business Transaction is
1996 created if the CONTEXT message for the existing Business Transaction is related to the
1997 BEGIN. In this case, the Factory is responsible for enrolling the new Composer or
1998 Coordinator as an Inferior of the Superior identified in that CONTEXT.
1999

2000 Note – This specification does not provide a standardised means to
2001 determine which of the Inferiors of a sub-Composer are in its confirm set.
2002 This is considered part of the application:inferior relationship.

2003
2004 The forms BEGIN/cohesion and BEGIN/atom refer to BEGIN with “transaction₋type”
2005 having the corresponding value.

2006
2007 Types of FAULT possible (sent to “reply₋address”)

2008
2009 **General**
2010 WrongState - only issued if there is a related CONTEXT, and the
2011 Superior identified by the CONTEXT is in the wrong state to enrol new
2012 Inferiors

2013
2014 **BEGUN**

2015
2016 BEGUN is a reply to BEGIN. There is always a related CONTEXT, which is the CONTEXT
2017 for the new business transaction.

Parameter	Type
target ₋ address	BTP address
address-as-decider	Set of BTP addresses

address-as-inferior	Set of BTP addresses
transaction-identifier	Identifier
inferior-identifier	Identifier
qualifiers	List of qualifiers

2019

2020

2021

2022

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

2023

2024

2025

2026

2027

2028

address-as-decider for a top-most transaction (no CONTEXT related to the BEGIN), this is the address to which PREPARE_INFERIORS, CONFIRM_TRANSACTION, CANCEL_TRANSACTION, CANCEL_INFERIORS and REQUEST_INFERIOR_STATUSES messages are to be sent; if a CONTEXT was related to the BEGIN this parameter is absent

2029

2030

2031

2032

2033

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

2034

2035

2036

2037

2038

2039

2040

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

2041

2042

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

2043

2044

2045

qualifiers standardised or other qualifiers.

2046

2047

2048

2049

2050

2051

At implementation option, the “address-as-decider” and/or “address-as-inferior” and the “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).

2052

No FAULT messages are issued on receiving BEGUN.

2053

2054

2055

PREPARE_INFERIORS

2056

2057

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

2058 PREPARED, RESIGN or CANCELLED to the Decider (Composer) on its relationships as
2059 Superior. If the inferiors-list parameter is absent, the request applies to all the inferiors; if the
2060 parameter is present, it applies only to the identified inferiors of the Decider (Composer).
2061

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

2062
2063 **target-address** the address to which the PREPARE_INFERIORS message is
2064 sent. This will be the decider-address from the BEGUN message.
2065

2066 **reply-address** the address of the Terminator sending the
2067 PREPARE_INFERIORS message.
2068

2069 **transaction identifier** identifies the Decider and will be the transaction-identifier
2070 from the BEGUN message.
2071

2072 **inferiors-list** defines which of the Inferiors of this Decider preparation is
2073 requested for, using the “inferior-identifiers” as on the ENROL received by the
2074 Decider (in its role as Superior). If this parameter is absent, the PREPARE
2075 applies to all Inferiors.
2076

2077 **qualifiers** standardised or other qualifiers.
2078
2079

2080 For all Inferiors identified in the inferiors-list parameter (all Inferiors if the parameter is
2081 absent), from which none of PREPARED, CANCELLED or RESIGNED has been received,
2082 the Decider shall issue PREPARE. It will reply to the Terminator, using the “reply-address”
2083 on the PREPARE_INFERIORS message, sending an INFERIOR_STATUSES message
2084 giving the status of the Inferiors identified on the inferiors-list parameter (all of them if the
2085 parameter was absent).
2086

2087 Types of FAULT possible (sent to Superior address)
2088

2089 **General**

2090 **InvalidDecider** – if Decider address is unknown

2091 **UnknownTransaction** – if the transaction-identifier is unknown

2092 **InvalidInferior** – if an inferior-handle on the inferiors-list is unknown

2093 **WrongState** – if a CONFIRM_TRANSACTION or
2094 CANCEL_TRANSACTION has already been received by this
2095 Composer.

2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107

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

CONFIRM_TRANSACTION

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

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

2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131

target-address the address to which the CONFIRM_TRANSACTION message is sent. This will be the 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 CONFIRM_COMPLETE or CANCEL_COMPLETE as soon as the decision for the transaction is known.

qualifiers standardised or other qualifiers.

2132
2133 If the “inferiors-list” parameter is present, the Inferiors identified shall be the “confirm-set” of
2134 the Cohesion. If the parameter is absent and the business transaction is a Cohesion, the
2135 “confirm-set” shall be all remaining Inferiors. If the business transaction is an Atom, the
2136 “confirm-set” is automatically all the Inferiors.
2137

2138 Any Inferiors from which RESIGN is received are not counted in the confirm-set.
2139

2140 If, for each of the Inferiors in the confirm-set, PREPARE has not been sent and PREPARED
2141 has not been received, PREPARE shall be issued to that Inferior.
2142

2143 NOTE -- If PREPARE has been sent but PREPARED not yet received from
2144 an Inferior in the confirm-set, it is an implementation option whether and
2145 when to re-send PREPARE. The Superior implementation may choose to re-
2146 send PREPARE if there are indications that the earlier PREPARE was not
2147 delivered.

2148
2149 A confirm decision may be made only if PREPARED has been received from all Inferiors in
2150 the “confirm-set”. The making of the decision shall be persistent (and if it is not possible to
2151 persist the decision, it is not made). If there is only one remaining Inferior in the “confirm
2152 set” and PREPARE has not been sent to it, CONFIRM_ONE_PHASE may be sent to it.
2153
2154

2155 All remaining Inferiors that are not in the confirm set shall be cancelled.
2156

2157 If a confirm decision is made and “report-hazard” was “false”, a CONFIRM_COMPLETE
2158 message shall be sent to the “reply-address”.
2159

2160 If a cancel decision is made and “report-hazard” was “false”, a CANCEL_COMPLETE
2161 message shall be sent to the “reply-address”.
2162

2163 If “report-hazard” was “true” and any HAZARD or contradictory message was received (i.e.
2164 CANCELLED from an Inferior in the confirm-set or CONFIRMED from an Inferior not in
2165 the confirm-set), an INFERIOR_STATUSES reporting the status for all Inferiors shall be sent
2166 to the “reply-address”.
2167

2168 Types of FAULT possible (sent to “reply-address”)

2169
2170 *General*

2171 *InvalidDecider* – if Decider address is unknown

2172 *UnknownTransaction* – if the transaction-identifier is unknown

2173 *InvalidInferior* – if an inferior handle in the inferiors-list is unknown

2174 *WrongState* – if a CANCEL_TRANSACTION has already been
2175 received .
2176

2177 The form CONFIRM_TRANSACTION/all refers to a CONFIRM_TRANSACTION message
 2178 where the “inferiors-list” parameter is absent. The form
 2179 CONFIRM_TRANSACTION/specific refers to a CONFIRM_TRANSACTION message
 2180 where the “inferiors-list” parameter is present.

2181
 2182
 2183
 2184
 2185
 2186
 2187
 2188

TRANSACTION_CONFIRMED

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

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

2189
 2190
 2191
 2192
 2193
 2194
 2195
 2196
 2197
 2198

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 address-as-decider)

2200
 2201
 2202
 2203
 2204

General

InvalidTerminator – if Terminator address is unknown

UnknownTransaction – if the transaction-identifier is unknown

CANCEL_TRANSACTION

2205
 2206
 2207
 2208
 2209

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

2210 qualifiers List of qualifiers

2211 **target-address** the address to which the CANCEL_TRANSACTION message
 2212 is sent. This will be the decider-address from the BEGUN message.

2213

2214 **reply-address** the address of the Terminator sending the
 2215 CANCEL_TRANSACTION message.

2216

2217 **transaction-identifier** identifies the Decider and will be the transaction-
 2218 identifier from the BEGUN message.

2219

2220 **report hazard** Defines whether the Terminator wishes to be informed of hazard
 2221 events and contradictory decisions within the business transaction. If “report
 2222 hazard” is “true”, the receiver will wait until responses (CONFIRMED,
 2223 CANCELLED or HAZARD) have been received from all of its inferiors,
 2224 ensuring that any hazard events are reported. If “report hazard” is “false”, the
 2225 Decider will reply with TRANSACTION_CANCELLED immediately.

2226

2227 **qualifiers** standardised or other qualifiers.

2228

2229 The business transaction is cancelled – this is propagated to any remaining Inferiors by
 2230 issuing CANCEL to them. No more Inferiors will be permitted to enrol.

2231 Types of FAULT possible (sent to Superior address)

2232 *General*

2233 *InvalidDecider* – if Decider address is unknown

2234 *UnknownTransaction* – if the transaction-identifier is unknown

2235 *WrongState* – if a CONFIRM_TRANSACTION has been received by
 2236 this Composer.

2237

2238

2239

2240

2241 **CANCEL_INFERIORS**

2242 Sent by a Terminator to a Decider, but only if is a Cohesion Composer, at any time before
 2243 CONFIRM_TRANSACTION or CANCEL_TRANSACTION has been sent.

2244

2245

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

2246
2247 **target-address** the address to which the CANCEL_TRANSACTION message
2248 is sent. This will be the decider-address from the BEGUN message.
2249
2250 **reply-address** the address of the Terminator sending the
2251 CANCEL_TRANSACTION message.
2252
2253 **transaction-identifier** identifies the Decider and will be the transaction-
2254 identifier from the BEGUN message.
2255
2256 **inferiors-list** defines which of the Inferiors of this Decider are to be cancelled,
2257 using the “inferior-identifiers” as on the ENROL received by the Decider (in its
2258 role as Superior).
2259
2260 **qualifiers** standardised or other qualifiers.

2261
2262
2263 Only the Inferiors identified in the inferiors-list are to be cancelled. Any other inferiors are
2264 unaffected by a CANCEL_INFERIORS. Further Inferiors may be enrolled.
2265

2266 Note – A CANCEL_INFERIORS all of the currently enrolled Inferiors will
2267 leave the cohesion ‘empty’, but permitted to continue with new Inferiors, if
2268 any enrol.

2269
2270 Types of FAULT possible (sent to Superior address)

2271 **General**
2272 **InvalidDecider** – if Decider address is unknown
2273 **UnknownTransaction** – if the transaction-identifier is unknown
2274 **InvalidInferior** – if an inferior-handle on the inferiors-list is unknown
2275 **WrongState** – if a CONFIRM_TRANSACTION or
2276 CANCEL_TRANSACTION has been received by this Composer.
2277
2278
2279

2280 2281 TRANSACTION_CANCELLED

2282
2283 A Decider sends TRANSACTION_CANCELLED to a Terminator in reply to
2284 CANCEL_TRANSACTION or in reply to CONFIRM_TRANSACTION if the Decider
2285 decided to cancel. In both cases, TRANSACTION_CANCELLED is used only if all Inferiors
2286 cancelled without reporting hazards or the CANCEL_TRANSACTION or
2287 CONFIRM_TRANSACTION had a “report-hazard” value of “false”.
2288

Parameter

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

2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314

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.

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 address-as-decider)

General

InvalidTerminator – if Terminator address is unknown

UnknownTransaction – if the transaction-identifier is unknown

REQUEST_INFERIOR_STATUSES

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

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

2315
2316
2317
2318
2319
2320
2321
2322

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

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

2323
 2324 **target-identifier** identifies the transaction (or transaction tree node) **within the**
 2325 **scope of the target address**. When the message is used to a Decider, this will be
 2326 the transaction-identifier from the BEGUN message. Otherwise it will be the
 2327 superior-identifier from a CONTEXT or an inferior-identifier from an ENROL
 2328 message.
 2329
 2330 **inferiors-list** defines which inferiors enrolled with the target are to be included
 2331 in the INFERIOR_STATUSES, using the “inferior-identifiers” as on the ENROL
 2332 received by the Decider (in its role as Superior). If the list is absent, the status of
 2333 all enrolled Inferiors will be reported.
 2334
 2335 **qualifiers** standardised or other qualifiers.

2336
 2337 Types of FAULT possible (sent to reply-address)
 2338

2339 *General*

2340 **StatusRefused** – if the receiver is not prepared to report its status to the
 2341 sender of this message. This **“fault-type”** shall not be issued when a Decider
 2342 receives REQUES_STATUSES from the Terminator.

2343 **UnknownTransaction** – if the transaction-identifier is unknown
 2344
 2345

2346 The form REQUEST_INFERIOR_STATUSES/all refers to a REQUEST_STATUS with the
 2347 inferiors-list absent. The form REQUEST_INFERIOR_STATUS/specific refers to a
 2348 REQUEST_INFERIOR_STATUS with the inferiors-list present.
 2349

2350 **INFERIOR_STATUSES**

2351
 2352 Sent by a Decider to report the status of all or some of its inferiors in response to a
 2353 REQUEST_INFERIOR_STATUSES, PREPARE_INFERIORS, CANCEL_INFERIORS,
 2354 CANCEL_TRANSACTION with “report-hazard” value of “true” and
 2355 CONFIRM_TRANSACTION with “report-hazard” value of “true”. It is also used by any
 2356 actor in response to a received REQUEST_INFERIOR_STATUSES to report the status of
 2357 inferiors, if there are any.
 2358

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

2359
 2360 **target-address** the address to which the INFERIOR_STATUSES is sent. This
 2361 will be the **“reply-address”** on the received message

2362
2363
2364
2365
2366
2367
2368

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).

2369
2370
2371
2372

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)

2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392

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

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

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

General

InvalidTerminator – if Terminator address is unknown

UnknownTransaction – if the transaction-identifier is unknown

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.

2403
2404
2405
2406
2407
2408
2409
2410

2411 **target-address:** the “target-address” is that of the application message. It is not required
2412 that the application address be a BTP address (in particular, there is no BTP-defined
2413 “additional information” field – the application protocol (and its binding) may or may not
2414 have a similar construct).

2415
2416 There may be multiple application messages related to a single CONTEXT message. All
2417 the application messages so related are deemed to be part of the business transaction
2418 identified by the CONTEXT. This specification does not imply any further relatedness
2419 among the application messages themselves (though the application might).

2420
2421 The actor that sends the group shall retain knowledge of the Superior address in the
2422 CONTEXT. If the CONTEXT is a CONTEXT/atom, the actor shall also keep track of
2423 transmitted CONTEXTs for which no CONTEXT_REPLY has been received.

2424
2425 If the CONTEXT is a CONTEXT/atom, the actor receiving the CONTEXT shall ensure
2426 that a CONTEXT_REPLY message is sent back to the “reply-address” of the
2427 CONTEXT with the appropriate completion status.

2428

2429 Note – The representation of the relation between CONTEXT and one or
2430 more application messages depends on the binding to the carrier protocol. It
2431 is not necessary that the CONTEXT and application messages be closely
2432 associated “on the wire” (or even sent on the same connection) – some kind
2433 of referencing mechanism may be used.

2434

2435 CONTEXT_REPLY & ENROL

2436

2437 **Meaning:** the enrolment of the Inferior identified in the ENROL is to be performed with
2438 the Superior identified in the CONTEXT message this CONTEXT_REPLY is replying
2439 to. If the “completion-status” of CONTEXT_REPLY is “related”, failure of this
2440 enrolment shall prevent the confirmation of the business transaction.

2441

2442 **target-address:** the “target-address” is that of the CONTEXT_REPLY. This will be the
2443 “reply-address” of the CONTEXT message (in many cases, including request/reply
2444 application exchanges, this address will usually be implicit).

2445

2446 The “target-address” of the ENROL message is omitted.

2447

2448 The actor receiving the related group will use the retained Superior address from the
2449 CONTEXT sent earlier to forward the ENROL. When doing so, it changes the ENROL to
2450 ask for a response (if it was an ENROL/no-rsp-req) and supplies its own address as the
2451 “reply-address”, remembering the original “reply-address” if there was one.

2452

2453 If ENROLLED is received and the original received ENROL was ENROL/rsp-req, the
2454 ENROLLED is forwarded back to the original “reply-address”.

2455

2456 If this attempt fails (i.e. ENROLLED is not received), and the “completion-status” of the
2457 CONTEXT_REPLY was “related”, the actor is required to ensure that the Superior does
2458 not proceed to confirmation. How this is achieved is an implementation option, but must
2459 take account of the possibility that direct communication with the Superior may fail. (One
2460 method is to prevent CONFIRM_TRANSACTION being sent to the Superior (in its role
2461 as Decider); another is to enrol as another Inferior before sending the original CONTEXT
2462 out with an application message). If the Superior is a sub-coordinator or sub-composer,
2463 an enrolment failure must ensure the sub-coordinator does not send PREPARED to its
2464 own Superior.

2465
2466 If the actor receiving the related group is also the Superior (i.e. it has the same binding
2467 address), the explicit forwarding of the ENROL is not required, but the resultant effect –
2468 that if enrolment fails the Superior does not confirm or issue PREPARED – shall be the
2469 same.

2470
2471 A CONTEXT_REPLY & ENROL group may contain multiple ENROL messages, for
2472 several Inferiors. Each ENROL shall be forwarded and an ENROLLED reply received
2473 before the Superior is allowed to confirm if the “completion-status” in the
2474 CONTEXT_REPLY was “related”.

2475
2476 When the group is constructed, if the CONTEXT had “superior-type” value of “atom”,
2477 the “completion-status” of the CONTEXT_REPLY shall be “related”. If the “superior-
2478 type” was “cohesive”, the “completion-status” shall be “completed” or “related” (as
2479 required by the application). If the value is “completed”, the actor receiving the group
2480 shall forward the ENROLs, but is not required to (though it may) prevent confirmation.

2481 2482 **CONTEXT_REPLY (& ENROL) & PREPARED / & CANCELLED**

2483
2484 This combination is characterised by a related CONTEXT_REPLY and either or both of
2485 PREPARED and CANCELLED, with or without ENROL.

2486
2487 **Meaning:** If ENROL is present, the meaning and required processing is the same as for
2488 CONTEXT_REPLY & ENROL. The PREPARED or CANCELLED message(s) are
2489 forwarded to the Superior identified in the CONTEXT message this CONTEXT_REPLY
2490 is replying to.
2491

2492 Note – the combination of CONTEXT_REPLY & ENROL & CANCELLED
2493 may be used to force cancellation of an atom

2494
2495 **target-address:** the “target-address” is that of the CONTEXT_REPLY. This will be the
2496 “reply-address” of the CONTEXT message (in many cases, including request/reply
2497 application exchanges, this address will usually be implicit).

2498
2499 The “target-address” of the PREPARED and CANCELLED message is omitted – they
2500 will be sent to the Superior identified in the earlier CONTEXT message.

2501
2502 The actor receiving the group forwards the PREPARED or CANCELLED message to the
2503 Superior in as for an ENROL, using the retained Superior address from the CONTEXT
2504 sent earlier, except there is no reply required from the Superior.
2505
2506 If (as is usual) an ENROL and PREPARED or CANCELLED message are for the same
2507 Inferior, the ENROL shall be sent first, but the actor need not wait for the ENROLLED to
2508 come back before sending the PREPARED or CANCELLED (so an
2509 ENROL+PREPARED bundle from this actor to the Superior could be used).
2510
2511 The group can contain multiple ENROL, PREPARED and CANCELLED messages.
2512 Each PREPARED and CANCELLED message will be for a different Inferior.. There is
2513 no constraint on the order of their forwarding, except that ENROL and PREPARED or
2514 CANCELLED for the same Inferior shall be delivered to the Superior in the order
2515 ENROL first, followed by the other message for that Inferior.
2516
2517
2518
2519 **CONTEXT_REPLY & ENROL & application message (& PREPARED)**
2520
2521 This combination is characterised by a related CONTEXT_REPLY, ENROL and an
2522 application message. PREPARED may or may not be present in the related group.
2523
2524 **Meaning:** the relation between the BTP messages is as for the preceding groups, The
2525 transmission of the application message (and application effects implied by its
2526 transmission) has been associated with the Inferior identified by the ENROL and will be
2527 subject to the outcome delivered to that Inferior.
2528
2529 **target-address:** the “target-address” of the group is the “target-address” of the
2530 CONTEXT_REPLY which shall also be the “target-address” of the application message.
2531 The ENROL and PREPARED messages do not contain their “target-address”
2532 [parameterses](#).
2533
2534 The processing of ENROL and PREPARED messages is the same as for the previous
2535 groups.
2536
2537 This group can be used when participation in business transaction (normally a cohesion),
2538 is initiated by the service (Inferior) side, which fetches or acquires the CONTEXT, with
2539 some associated application semantic, performs some work for the transaction and sends
2540 an application message with a related ENROL. The CONTEXT_REPLY allows the
2541 addressing of the application (and the CONTEXT_REPLY) to be distinct from that of the
2542 Superior.
2543
2544 The actor receiving the group may associate the “inferior-identifier” received on the
2545 ENROL with the application message in a manner that is visible to the application
2546 receiving the message (e.g. for subsequent use in Terminator:Decider exchanges).
2547

2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590

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: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

2591

2592 **Timelimit** indicates the maximum (further) duration, expressed as whole seconds from the
2593 time of transmission of the containing CONTEXT, of the active phase of the business
2594 transaction.

2595
2596 **Inferior timeout**

2597
2598 This qualifier allows an Inferior to limit the duration of its “promise”, when sending
2599 PREPARED, that it will maintain the ability to confirm or cancel the effects of all associated
2600 operations. Without this qualifier, an Inferior is expected to retain the ability to confirm or
2601 cancel indefinitely. If the timeout does expire, the Inferior is released from its promise and
2602 can apply the decision indicated in the qualifier.

2603
2604 It should be noted that BTP recognises the possibility that an Inferior may be forced to apply
2605 a confirm or cancel decision before the CONFIRM or CANCEL is received and before this
2606 timeout expires (or if this qualifier is not used). Such a decision is termed a heuristic decision,
2607 and (as with other transaction mechanisms), is considered to be an exceptional event. As with
2608 heuristic decisions, the taking of an autonomous decision by a Inferior **subsequent** to the
2609 expiry of this timeout, is liable to cause contradictory decisions across the business
2610 transaction. BTP ensures that at least the occurrence of such a contradiction will be
2611 (eventually) reported to the Superior of the business transaction. BTP treats “true” heuristic
2612 decisions and autonomous decisions after timeout the same way – in fact, the expiry in this
2613 timeout does not cause a qualitative (state table) change in what can happen, but rather a step
2614 change in the probability that it will.

2615
2616 The expiry of the timeout does not strictly require that the Inferior immediately invokes the
2617 intended decision, only that is at liberty to do so. An implementation may choose to only
2618 apply the decision if there is contention for the underlying resource, for example.
2619 Nevertheless, Superiors are recommended to avoid relying on this and ensure decisions for
2620 the business transaction are made before these timeouts expire (and allow a margin of error
2621 for network latency etc.).

2622
2623 The qualifier may be present on a PREPARED message. If the PREPARED message has the
2624 “default_is_cancel” parameter “true”, then the “IntendedDecision” field of this qualifier shall
2625 have the value “cancel”.

2626
2627 The “Qualifier name” shall be “inferior-timeout”.

2628
2629 The “Content” shall contain the following fields:
2630

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

2631
2632 **Timeout** indicates how long, expressed as whole seconds from the time of transmission of the
2633 carrying message, the Inferior intends to maintain its ability to either confirm or cancel the
2634 effects of the associated operations, as ordered by the receiving Superior.

2635
2636
2637
2638
2639
2640
2641
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

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.)

2679 The qualifier may be present on BEGIN, ENROL and in the “qualifiers” field of a Status-item
2680 in INFERIOR_STATUSES. It is present on BEGIN only if there is a related CONTEXT; if
2681 present, the same qualifier value **should** be included in the consequent ENROL. If
2682 INFERIOR_STATUSES includes a Status-item for an Inferior whose ENROL had an
2683 inferior-name qualifier, the same qualifier value **should** be included in the Status-item.
2684

2685 The “Qualifier -name” shall be “inferior-name”

2686

2687 The “Content” shall contain the following fields:

2688

Content field	Type
inferior-name	String

2689

2690 **Inferior name** the name assigned to the enrolling Inferior.

2691

2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737

State Tables

Explanation of the state tables

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

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

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

Status queries

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

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

Decision events

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

2738 business transaction and on features of the implementation (e.g. making of a persistent record
2739 of the decision means that the information will survive at least some failures that otherwise
2740 lose state information, but the level of survival depends on the purpose of the
2741 implementation). [Table 2](#)~~Table 2~~~~Table 2~~~~Table 2~~ and [Table 3](#)~~Table 3~~~~Table 3~~~~Table 3~~ define the
2742 decision events.

2743
2744 In some cases, an implementation may not need to make an active change to have a persistent
2745 record of a decision, provided that the implementation will restore itself to the appropriate
2746 state on recovery. For example, an (inferior) implementation that “decided to be prepared”,
2747 and recorded a timeout (to cancel) in the persistent information for that decision (signalled via
2748 the appropriate qualifier on PREPARED), could treat the presence of an expired record as a
2749 record of “decide to cancel autonomously”, provided it always updated such a record as part
2750 of the “apply ordered confirmation” decision event.

2751
2752 The Superior event “decide to prepare” is considered semi-persistent. Since the sending of
2753 PREPARE indicates that the application exchange (to associate operations with the Inferior)
2754 is complete, it is not meaningful for the Superior:Inferior relationship to revert to an earlier
2755 state corresponding to an incomplete application exchange. However, implementations are
2756 not required to make the sending of PREPARE persistent in terms of recovery – a Superior
2757 that experiences failure after sending PREPARE may, on recovery, have no information
2758 about the transaction, in which case it is considered to be in the completed state (Z), which
2759 will imply the cancellation of the Inferior and its associated operations.

2760
2761 Where a Superior is itself an Inferior (to another Superior entity), in a hierarchic tree, its
2762 “decide to confirm” and “decide to cancel” decisions will in fact be the receipt of a
2763 CONFIRM or CANCEL instruction from its own Superior, without necessary change of local
2764 persistent information (which would combine both superior and inferior information, pointing
2765 both up and down the tree).

2766
2767

2768 **Disruptions – failure events**

2769
2770 Failure events are modelled as “disruption”. A failure and the subsequent recovery will (or
2771 may) cause a change of state. The disruption events in the state tables model different extents
2772 of loss of state information. An implementation is not required to exhibit all the possible
2773 disruption events, but it is not allowed to exhibit state transitions that do not correspond to a
2774 possible disruption.

2775
2776 In addition to the disruption events in the tables, there is an implicit “disruption 0” event,
2777 which involves possible interruption of service and loss of messages in transit, but no change
2778 of state (either because no state information was lost, or because recovery from persistent
2779 information restores the implementation to the same state). The “disruption 0” event would
2780 typically be an appropriate abstraction for a communication failure.

2781
2782 **Invalid cells and assumptions of the communication mechanism**
2783

2784 The empty cells in state table represent events that cannot happen. For events corresponding
2785 to sending a message or any of the decision events, this prohibition is absolute – e.g. a
2786 conformant implementation in the Superior active state “B1” will not send CONFIRM. For
2787 events corresponding to receiving a message, the interpretation depends on the properties of
2788 the underlying communications mechanism.
2789

2790 For all communication mechanisms, it is assumed that

- 2791 a) the two directions of the Superior:Inferior communication are not synchronised –
2792 that is messages travelling in opposite directions can cross each other to any
2793 degree; any number of messages may be in transit in either direction; and
- 2794 b) messages may be lost arbitrarily
2795

2796 If the communication mechanisms guarantee ordered delivery (i.e. that messages, if delivered
2797 at all, are delivered to the receiver in the order they were sent) , then receipt of a message in a
2798 state where the corresponding cell is empty indicates that the far-side has sent a message out
2799 of order – a FAULT message with the `“fault_type”` `“WrongState”` can be returned.
2800

2801 If the communication mechanisms cannot guarantee ordered delivery, then messages received
2802 where the corresponding cell is empty should be ignored. Assuming the far-side is
2803 conformant, these messages can assumed to be “stale” and have been overtaken by messages
2804 sent later but already delivered. (If the far-side is non-conformant, there is a problem
2805 anyway).
2806

2807 **Meaning of state table events**

2808
2809 The tables in this section define the events (rows) in the state tables. [Table 1](#) ~~Table 1~~
2810 ~~Table 1~~ defines the events corresponding to sending or receiving BTP messages and the
2811 disruption events. [Table 2](#) ~~Table 2~~
2812 ~~Table 2~~ describes the decision events for an Inferior, [Table 3](#) ~~Table 3~~
2813 ~~Table 3~~ those for a Superior.

2814 The decision events for a Superior, defined in [Table 3](#) ~~Table 3~~
2815 ~~Table 3~~ cannot be specified without reference to other Inferiors to which it is Superior and to its relation with
2816 the application or other entity that (acting ultimately on behalf of the application) drives it.
2817

2818 The term “remaining Inferiors” refers to any actors to which this endpoint is Superior and
2819 which are to be treated as an atomic decision unit with (and thus including) the Inferior on
2820 this relationship. If the CONTEXT for this Superior:Inferior relationship had a `“superior_`
2821 `type”` of `“atom”`, this will be all Inferiors established with same Superior address and
2822 `“superior_`
2823 `identifier”` except those from which RESIGN has been received. If the CONTEXT
2824 had `“superior_`
2825 `type”` of `“cohesion”`, the “remaining Inferiors” excludes any that it has been
2826 determined will be cancelled, as well as any that have resigned – in other words it includes
2827 only those for which a confirm decision is still possible or has been made. The determination
2828 of exactly which Inferiors are “remaining Inferiors” in a cohesion is determined, in some
2829 way, by the application. The term “Other remaining Inferiors” excludes this Inferior on this
relationship. A Superior with a single Inferior will have no “other remaining Inferiors”.

2830 In order to ensure that the confirmation decision **is** delivered to all remaining Inferiors,
 2831 despite failures, the Superior must persistently record which these Inferiors are (i.e. their
 2832 addresses and identifiers). It must also either record that the decision is confirm, or ensure
 2833 that the confirm decision (if there is one) is persistently recorded somewhere else, and that it
 2834 will be told about it. This latter would apply if the Superior were also BTP Inferior to another
 2835 entity which persisted a confirm decision (or recursively deferred it still higher). However,
 2836 since there is no requirement that the Superior be also a BTP Inferior to any other entity, the
 2837 behaviour of asking another entity to make (and persist) the confirm decision is termed
 2838 "offering confirmation" - the Superior offers the possible confirmation of itself, and its
 2839 remaining Inferiors to some other entity. If that entity (or something higher up) then does
 2840 make and persist a confirm decision, the Superior is "instructed to confirm" (which is
 2841 equivalent BTP CONFIRM).

2842
 2843 The application, or an entity acting indirectly on behalf of the application, may request a
 2844 Superior to prepare an Inferior (or all Inferiors). This typically implies that there will be no
 2845 more operations associated with the Inferior. Following a request to prepare all remaining
 2846 Inferiors, the Superior may offer confirmation to the entity that requested the prepare. (If the
 2847 Superior is also a BTP Inferior, its superior can be considered an entity acting on behalf of the
 2848 application.)

2849
 2850 The application, or an entity acting indirectly on behalf of the application, may also request
 2851 confirmation. This means the Superior is to attempt to make and persist a confirm decision
 2852 itself, rather than offer confirmation.

2853
 2854

2855

Table 1 : send, receive and disruption events

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

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

2856

2857

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

Event name	Meaning
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;
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)

2858

2859

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

Event name	Meaning
	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 • 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)

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

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

2892

2893

2894

2895

2896

2897

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										F1
send CONFIR M										
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								
deci de to confi rm					F1	K1		
deci de to cancel					L1	G4		
remove persi stent i nformati on								
record contradi cti on							R1	R1
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		

2900

2901

2901

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								

2902

2903

2903

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		

2904

2905

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
disruption I disruption II disruption III		z	z	z	z b1			e1	e2

Table 11: Inferior state table – cancellation and contradiction

	g1	g2	h1	h2	j1	j2	k1	k2	l1	l2
send ENROL/rsp-req send ENROL/no-rsp-req send RESIGN/rsp-req send RESIGN/no-rsp-req send PREPARED send PREPARED/cancel send CONFIRMED/auto send CONFIRMED/response send CANCELLED send HAZARD			h1		j1		k1		l1	
send INF_STATE/active/y send INF_STATE/active send INF_STATE/unknown										
receive ENROLLED receive RESIGNED receive PREPARE receive CONFIRM_ONE_PHASE receive CONFIRM receive CANCEL receive CONTRADICTION	g1	g2	h1 h1 h2	h2	j1 j1 k1	j2	k1	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		j1 j1 j1 j1	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	n1		m1		z		z		z
disruption I disruption II disruption III	e1	e2	h1		j1		j1	k1 j1	h1	l1 h1

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

	m1	n1	p1	p2	q1
send ENROL/rsp-req send ENROL/no-rsp-req send RESI GN/rsp-req send RESI GN/no-rsp-req send PREPARED send PREPARED/cancel send CONFIR MED/auto send CONFIR MED/response send CANCELLED send HAZARD	z	z	p1	p2	q1
send INF_STATE/active/y send INF_STATE/active send INF_STATE/unknown					
recei ve ENROLLED recei ve RESI GNED recei ve PREPARE recei ve CONFIR M_ONE_PHASE recei ve CONFIR M recei ve CANCEL recei ve CONTRADI CTI ON	m1	n1	p1 s5 z	p2 s5 z	q1 s6 q1 q1 z
recei ve SUP_STATE/active/y recei ve SUP_STATE/active recei ve SUP_STATE/prepared-rcvd/y recei ve SUP_STATE/prepared-rcvd recei ve SUP_STATE/unknown		z	p1 p1 p1	p2 p2 p2	q1 q1 q1 q1 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 apply 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		

2910

2911

2911

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	s2		s3 s4			s6
di srupti on I di srupti on II di srupti on III	e1	z		z	z	

2912

2913

2913

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

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

2914

2915

2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960

Failure Recovery

Types of failure

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

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

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

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

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

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

2961 After recovery from node failure, the implementation behaves much as if a communication
2962 failure had occurred.

2963

2964 Persistent information

2965

2966 BTP requires that some decision events are persisted – that information recording an
2967 Inferior’s decision to be prepared, a Superior’s decision to confirm and an Inferior’s
2968 autonomous decision survive failure. Making the first two decisions persistent ensures that a
2969 consistent decision can be reached for the business transaction and that it is delivered to all
2970 involved nodes. Requiring an Inferior’s autonomous decision to be persistent allows BTP to
2971 ensure that, if this decision is contradictory (i.e. opposite to the decision at the Superior), the
2972 contradiction will be reported to the Superior, despite failures.

2973

2974 BTP also permits, but does not require, recovery of the Superior:Inferior relationship in the
2975 active state (unlike many transaction protocols, where a communication or endpoint failure in
2976 active state would invariably cause rollback of the transaction). Recovery in the active state
2977 may require that the application exchange is resynchronised as well – BTP does not directly
2978 support this, but does allow continuation of the business transaction as such. In the state
2979 tables, from some states, there are several levels of disruption, distinguished by which state
2980 the implementation transits to – this represents the survival of different extents of state
2981 information over failure and recovery. The different levels of disruption describe legitimate
2982 states for the endpoint to be in after it has recovered – **they do not require that all
2983 implementations are able to exhibit the appropriate partial loss of state information.**

2984 The absence of a destination state for the disruption events means that such a transition is not
2985 legitimate – thus, for example, an Inferior that has decided to be prepared will always recover
2986 to the same state, by virtue of the information persisted in the “decide to be prepared” event.

2987

2988 Apart from the (optional) recovery in active state, BTP follows the well-known presume-
2989 abort model – it is only required that information be persisted when decisions are made (and
2990 not, e.g. on enrolment). This means that on recovery, one side may have persistent
2991 information but the other does not. This occurs when an Inferior has decided to be prepared
2992 but the Superior never confirmed (so the decision is “presumed” to be cancel), or because the
2993 Superior did confirm, and the Inferior applied the confirm, removed its persistent information
2994 but the acknowledgement (CONFIRMED) was never received by the Superior (or, at least, it
2995 still had the persistent information when the failure occurred).

2996

2997 Information to be persisted for an Inferior’s “decision to be prepared” must be sufficient to
2998 re-establish communication with the Superior, to apply a confirm decision and to apply a
2999 cancel decision. It will thus need to include

3000

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

3001

Superior address (as on CONTEXT)

3002

“superior-identifier” (as on CONTEXT)

3003

default-is-cancel value (as on PREPARED)

3004

3005 The information needed to apply confirm/cancel decisions will depend on the application and
3006 the associated operations. It may also normally be necessary to persist any qualifiers that

3007 were sent with the PREPARED message or application messages sent with the PREPARED,
3008 since the PREPARED message will be repeated if a failure occurs.

3009
3010 A Superior must record corresponding information to allow it to re-establish communication
3011 with the Inferior:

3012 Inferior address (as on ENROL)
3013 “inferior-identifier” (as on ENROL)

3014
3015 A Superior that is the Decider for the business transaction need only persist this information
3016 if it makes a decision to confirm (and this Inferior is in the confirm set, for a Cohesion). A
3017 Superior that is also an Inferior to some other entity (i.e. it is an intermediate in a tree, as
3018 atom in a cohesion, sub-coordinator or sub-composer) must persist this information as
3019 Superior (to this Inferior) as part of the persistent information of its decision to be prepared
3020 (as an Inferior). For such an entity, the “decision to confirm” as Superior is made when (and
3021 if) CONFIRM is received from its Superior or it makes an autonomous decision to confirm. If
3022 CONFIRM is received, the persistent information may be changed to show the confirm
3023 decision, but alternatively, the receipt of the CONFIRM can be treated as the decision itself.
3024 If the persistent information is left unchanged and there is a node failure, on recovery the
3025 entity (as an Inferior) will be in a prepared state, and will rediscover the confirm decision
3026 (using the recovery exchanges to its Superior) before propagating it to its Inferior(s).

3027
3028 After failure, an implementation may not be able to restore an endpoint to the appropriate
3029 state immediately – in particular, the necessary persistent information may be inaccessible,
3030 although the implementation can respond to received BTP messages. In such a case, a
3031 Superior may reply to any BTP message except INFERIOR_STATE/* (i.e. with a
3032 “reply-response-requested” value “false”) with SUPERIOR_STATE/inaccessible and an
3033 Inferior to any BTP message except SUPERIOR_STATE/* with
3034 “INFERIOR_STATE/inaccessible. Receipt of the *_STATE/inaccessible messages has no
3035 effect on the endpoint state.

3036 3037 **Redirection**

3038
3039 As described above, BTP uses the presume-abort model for recovery. A corollary of this is
3040 that there are cases where one side will attempt to re-establish communication when there is
3041 no persistent information for the relationship at the far-end. In such cases, it is important the
3042 side that is attempting recovery can distinguish between unsuccessful attempts to connect to
3043 the holder of the persistent information and when the information no longer exists. If the peer
3044 information does not exist, this side can draw conclusions and complete appropriately; if they
3045 merely fail to get through they are stuck in attempting recovery.

3046
3047 Two mechanisms are provided to make it possible that even when one side of a
3048 Superior:Inferior relationship has completed, that a message can eventually get through to
3049 something that can definitively report the status, distinguishing this case from a temporary
3050 inability to access the state of a continuing transaction element. The mechanisms are:
3051 o Address fields which provide a “callback address” can be a set of addresses,
3052 which are alternatives one of which is chosen as the “target-address” for the

3053 future message. If the sender of that message finds the address does not work,
3054 it can try a different alternative.
3055 o The REDIRECT message can be used to inform the peer that an address
3056 previously given is no longer valid and to supply a replacement address (or
3057 set of addresses). REDIRECT can be issued either as a response to receipt of
3058 a message or spontaneously.
3059

3060 The two mechanisms can be used in combination, with one or more of the original set of
3061 addresses just being a redirector, which does not itself ever have direct access to the state
3062 information for the transaction, but will respond to any message with an appropriate
3063 REDIRECT.
3064

3065 An alternative implementation approach is to have a single addressable entity that uses the
3066 same address for all transactions, distinguishing them by identifier, and which always
3067 recovers to use the same address. Such an implementation would not need to supply
3068 “backup” addresses (and would only use REDIRECT if it was being permanently migrated).
3069

3070 Terminator:Decider failures

3071
3072 BTP does not provide facilities or impose requirements on the recovery of
3073 Terminator:Decider relationships, other than allowing messages to be repeated. A Terminator
3074 may survive failures (by retaining knowledge of the Decider’s address and identifier), but this
3075 is an implementation option. Although a Decider (if it decides to confirm) will persist
3076 information about the confirm decision, it is not required, after failure, to remain accessible
3077 using the inferior address it offered to the Terminator. Any such recovery is an
3078 implementation option.
3079

3080 A Decider’s address (as returned on BEGUN) may be a set of addresses, allowing a failed
3081 Decider to be recovered at a different address.
3082

3083 A Decider has no way of initiating a call to a Terminator to ensure that it is still active, and
3084 thus no way of detecting that a Terminator has failed. To avoid a Decider waiting for ever for
3085 a CONFIRM_TRANSACTION that will never arrive, the standard qualifier “Transaction
3086 timelimit” can be used (by the Initiator) to inform the Decider when it can assume the
3087 Terminator will not issue CONFIRM_TRANSACTION and so it (the Decider) should initiate
3088 cancellation.
3089

3090 XML representation of Message Set

3091
3092 This section describes the syntax for BTP messages in XML. These XML messages represent
3093 a midpoint between the abstract messages and what actually gets sent on the wire.
3094

3095 All BTP related URIs have been created using Oasis URI conventions as specified in [RFC](#)
3096 [3121](#)
3097

3098 The XML Namespace for the BTP messages is urn:oasis:names:tc:BTP:xml
3099

3100 In addition to an XML schema, this specification uses an informal syntax to describe the
3101 structure of the BTP messages. The syntax appears as an XML instance, but the values
3102 contain data types instead of values. The following symbols are appended to some of the
3103 XML constructs: ? (zero or one), * (zero or more), + (one or more.) The absence of one of
3104 these symbols corresponds to "one and only one."
3105

3106 Addresses

3107
3108 As described in the “Abstract Message and Associated Contracts – Addresses” section, a BTP
3109 address comprises three parts, and for a “target-address” only the “additional information”
3110 field is inside the BTP messages. For all BTP messages whose abstract form includes a
3111 “target-address” parameter, the corresponding XML representation includes a “target-
3112 additional-information” element. This element may be omitted if it would be empty.
3113

3114 For other addresses, all three fields are represent, as in:

```
3115  
3116 <btp:some-address>  
3117   <btp:binding-name>...carrier binding URI...</btp:binding-name>  
3118   <btp:binding-address>...carrier specific  
3119   address...</btp:binding-address>  
3120   <btp:additional-information>...optional additional addressing  
3121   information...</btp:additional-information> ?  
3122 </btp:some-address>  
3123
```

3124
3125 A "published" address can be a set of <some-address>, which are alternatives which can be
3126 chosen by the peer (sender.) Multiple addresses are used in two cases: different bindings to
3127 same endpoint, or backup endpoints. In the former, the receiver of the message has the choice
3128 of which address to use (depending on which binding is preferable.) In the case where
3129 multiple addresses are used for redundancy, a priority attribute can be specified to help the
3130 receiver choose among the addresses- the address with the highest priority should be used,
3131 other things being equal. The priority is used as a hint and does not enforce any behaviour in
3132 the receiver of the message. Default priority is a value of 1.
3133

3134 Qualifiers

3135 The “Qualifier name” is used as the element name, within the namespace of the “Qualifier
3136 group”.

3137 Examples:

```
3138  
3139 <btpq:inferior-timeout  
3140   xmlns:btpq="urn:oasis:names:tc:BTP:qualifiers"  
3141   xmlns:btp="urn:oasis:names:tc:BTP:xml "  
3142   btp:must-be-understood="false"  
3143   btp:to-be-propagated="false">1800</btpq:inferior-timeout>  
3144  
3145 <auth:username  
3146   xmlns:auth="http://www.example.com/ns/auth"  
3147   xmlns:btp="urn:oasis:names:tc:BTP:xml "  
3148   btp:must-be-understood="true"
```

3149 `btp:to-be-propagated="true">jtauber</auth:username>`
3150
3151 Attributes must-be-understood **has default value “true”** and to-be-propagated has default
3152 value “false”.

3153 Identifiers

3154 Identifiers shall be URIs "
3155
3156
3157

3158 Note – Identifiers need to be globally unambiguous. Apart from their
3159 generation, the only operation the BTP implementations have to perform on
3160 identifiers is to match them.

3161 Message References

3162 Each BTP message has an optional id attribute to give it a unique identifier. An application
3163 can make use of those identifiers, but no processing is enforced.
3164
3165

3166 Messages

3167 CONTEXT

```
3170 <btp:context id?>  
3171 <btp:superior-address> +  
3172 ...address...  
3173 </btp:superior-address>  
3174 <btp:superior-identifier>...URI...</btp:superior-identifier>  
3175 <btp:reply-address> ?  
3176 ...address...  
3177 </btp:reply-address>  
3178 <btp:superior-type>cohesion|atom</btp:superior-type>  
3179 <btp:qualifiers> ?  
3180 ...qualifiers...  
3181 </btp:qualifiers>  
3182 </btp:context>
```

3183 CONTEXT_REPLY

```
3184  
3185  
3186 <btp:context-reply id?>  
3187 <btp:target-additional-information> ?  
3188 ...additional address information...  
3189 </btp:target-additional-information>  
3190  
3191 <btp:superior-identifier>...URI...</btp:superior-identifier>  
3192 <btp:completion-  
3193 status>completed|related|repudiated</btp:completion-status>  
3194 <btp:qualifiers> ?  
3195 ...qualifiers...  
3196 </btp:qualifiers>
```

3197 </btp:context-reply>

3198

3199

REQUEST_STATUS

3200

3201

```
<btp:request-status id?>
```

3202

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

3203

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

3204

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

3205

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

3206

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

3207

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

3208

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

3209

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

3210

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

3211

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

3212

```
</btp:request-status>
```

3213

3214

STATUS

3215

3216

```
<btp:status id?>
```

3217

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

3218

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

3219

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

3220

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

3221

3222

```
<btp:status-value>created|enrolling|active|resigning|
```

3223

```
resigned|preparing|prepared|
```

3224

```
confirming|confirmed|cancelling|cancelled|
```

3225

```
cancel-contradiction|confirm-contradiction|
```

3226

```
hazard|contradicted|unknown|inaccessible</btp:status-
```

3227

```
value>
```

3228

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

3229

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

3230

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

3231

```
</btp:status>
```

3232

3233

FAULT

3234

3235

```
<btp:fault id?>
```

3236

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

3237

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

3238

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

3239

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

3240

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

3241

```
<btp:fault-type>...fault type name...</btp:fault-type>
```

3242

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

3243

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

3244

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

3245

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

3246

```
</btp:fault>
```

3247

3248 The following fault type names are represented by simple strings, corresponding to the entries
3249 defined in the abstract message set:

- 3250
- 3251 o communication-failure
- 3252 o duplicate-inferior
- 3253 o general
- 3254 o invalid-decider
- 3255 o invalid-inferior
- 3256 o invalid-superior
- 3257 o status-refused
- 3258 o invalid-terminator
- 3259 o unknown-parameter
- 3260 o unknown-transaction
- 3261 o unsupported-qualifier
- 3262 o wrong-state

3263 Revisions of this specification may add other fault type names, which shall be simple strings
3264 of letters, numbers and hyphens. If other specifications define fault type names to be used
3265 with BTP, the names shall be URIs.
3266

3267
3268 Fault data can take on various forms:

3269
3270 Free text:

```
3271 <btp: fault-data>...string data...</btp: fault-data>
```

3272
3273 Identifier:

```
3274 <btp: fault-data>...URI...</btp: fault-data>
```

3275
3276
3277 Inferior Identity:

```
3280 <btp: fault-data>  
3281 <btp: inferior-address> +  
3282 ...address...  
3283 </btp: inferior-address>  
3284 <btp: inferior-identifier>...URI...</btp: inferior-identifier>  
3285 </btp: fault-data>
```

3286
3287
3288 **ENROL**

```
3289 <btp: enrol id?>  
3290 <btp: target-additional-information> ?  
3291 ...additional address information...  
3292 </btp: target-additional-information>  
3293 <btp: superior-identifier>...URI...</btp: superior-identifier>
```

```
3295 <ctp:replyresponse-requested>true|false</ctp:replyresponse-  
3296 requested>  
3297 <ctp:reply-address> ?  
3298 ...address...  
3299 </ctp:reply-address>  
3300 <ctp:inferior-address> +  
3301 ...address...  
3302 </ctp:inferior-address>  
3303 <ctp:inferior-identifier>...URI...</ctp:inferior-identifier>  
3304 <ctp:qualifiers> ?  
3305 ...qualifiers...  
3306 </ctp:qualifiers>  
3307 </ctp:enrol>
```

3308
3309

ENROLLED

```
3311  
3312 <ctp:enrolled id?>  
3313 <ctp:target-additional-information> ?  
3314 ...additional address information...  
3315 </ctp:target-additional-information>  
3316 <ctp:inferior-identifier>...URI...</ctp:inferior-identifier>  
3317 <ctp:qualifiers> ?  
3318 ...qualifiers...  
3319 </ctp:qualifiers>  
3320 </ctp:enrolled>
```

3321
3322

RESIGN

```
3324  
3325 <ctp:resign id?>  
3326 <ctp:target-additional-information> ?  
3327 ...additional address information...  
3328 </ctp:target-additional-information>  
3329 <ctp:superior-identifier>...URI...</ctp:superior-identifier>  
3330 <ctp:inferior-identifier>...URI...</ctp:inferior-identifier>  
3331 <ctp:response-requested>true|false</ctp:response-requested>  
3332 <ctp:qualifiers> ?  
3333 ...qualifiers...  
3334 </ctp:qualifiers>  
3335 </ctp:resign>
```

3336
3337

RESIGNED

```
3339  
3340 <ctp:resigned id?>  
3341 <ctp:target-additional-information> ?  
3342 ...additional address information...  
3343 </ctp:target-additional-information>  
3344 <ctp:inferior-identifier>...URI...</ctp:inferior-identifier>  
3345 <ctp:qualifiers> ?
```

3346 ...qualifiers...
3347 </btp:qualifiers>
3348 </btp:resigned>

3349
3350

PREPARE

3351
3352
3353 <btp:prepare id?>
3354 <btp:target-additional-information> ?
3355 ...additional address information...
3356 </btp:target-additional-information>
3357 <btp:inferior-identifier>...URI...</btp:inferior-identifier>
3358 <btp:qualifiers> ?
3359 ...qualifiers...
3360 </btp:qualifiers>
3361 </btp:prepare>

3362
3363

PREPARED

3364
3365
3366 <btp:prepared id?>
3367 <btp:target-additional-information> ?
3368 ...additional address information...
3369 </btp:target-additional-information>
3370 <btp:superior-identifier>...URI...</btp:superior-identifier>
3371 <btp:inferior-identifier>...URI...</btp:inferior-identifier>
3372 <btp:default-is-cancel>true|false</btp:default-is-cancel>
3373 <btp:qualifiers> ?
3374 ...qualifiers...
3375 </btp:qualifiers>
3376 </btp:prepared>

3377
3378

CONFIRM

3379
3380
3381 <btp:confirm id?>
3382 <btp:target-additional-information> ?
3383 ...additional address information...
3384 </btp:target-additional-information>
3385 <btp:inferior-identifier>...URI...</btp:inferior-identifier>
3386 <btp:qualifiers> ?
3387 ...qualifiers...
3388 </btp:qualifiers>
3389 </btp:confirm>

3390
3391

CONFIRMED

3392
3393
3394 <btp:confirmed id?>
3395 <btp:target-additional-information> ?
3396 ...additional address information...

```
3397 </btp:target-additional-information>
3398 <btp:superior-identifier>...URI...</btp:superior-identifier>
3399 <btp:inferior-identifier>...URI...</btp:inferior-identifier>
3400 <btp:confirmed-received>true|false</btp:confirmed-received>
3401 <btp:qualifiers> ?
3402   ...qualifiers...
3403 </btp:qualifiers>
3404 </btp:confirmed>
```

3405
3406

CANCEL

```
3407
3408
3409 <btp:cancel id?>
3410   <btp:target-additional-information> ?
3411     ...additional address information...
3412   </btp:target-additional-information>
3413   <btp:inferior-identifier>...URI...</btp:inferior-identifier>
3414   <btp:reply-address> ?
3415     ...address...
3416   </btp:reply-address>
3417   <btp:qualifiers> ?
3418     ...qualifiers...
3419   </btp:qualifiers>
3420 </btp:cancel>
```

3421
3422

CANCELLED

```
3423
3424
3425 <btp:cancelled id?>
3426   <btp:target-additional-information> ?
3427     ...additional address information...
3428   </btp:target-additional-information>
3429   <btp:superior-identifier>...URI...</btp:superior-identifier>
3430
3431   <btp:inferior-identifier>...URI...</btp:inferior-identifier> ?
3432   <btp:qualifiers> ?
3433     ...qualifiers...
3434   </btp:qualifiers>
3435 </btp:cancelled>
```

3436
3437

CONFIRM_ONE_PHASE

```
3438
3439
3440 <btp:confirm-one-phase id?>
3441   <btp:target-additional-information> ?
3442     ...additional address information...
3443   </btp:target-additional-information>
3444   <btp:inferior-identifier>...URI...</btp:inferior-identifier>
3445   <btp:report-hazard>true|false</btp:report-hazard>
3446   <btp:qualifiers> ?
3447     ...qualifiers...
```

3448 </btp:qualifiers>
3449 </btp:confirm-one-phase>

HAZARD

3451
3452
3453 <btp:hazard id?>
3454 <btp:target-additional-information> ?
3455 ...additional address information...
3456 </btp:target-additional-information>
3457 <btp:superior-identifier>...URI...</btp:superior-identifier>
3458
3459 <btp:inferior-identifier>...URI...</btp:inferior-identifier>
3460 <btp:level>mixed|possible</btp:level>
3461 <btp:qualifiers> ?
3462 ...qualifiers...
3463 </btp:qualifiers>
3464 </btp:hazard>

CONTRADICTION

3465
3466
3467
3468
3469 <btp:contradiction id?>
3470 <btp:target-additional-information> ?
3471 ...additional address information...
3472 </btp:target-additional-information>
3473 <btp:inferior-identifier>...URI...</btp:inferior-identifier>
3474 <btp:qualifiers> ?
3475 ...qualifiers...
3476 </btp:qualifiers>
3477 </btp:contradiction>

SUPERIOR_STATE

3478
3479
3480
3481
3482 <btp:superior-state id?>
3483 <btp:target-additional-information> ?
3484 ...additional address information...
3485 </btp:target-additional-information>
3486 <btp:inferior-identifier>...URI...</btp:inferior-identifier>
3487 <btp:status>active|prepared-
3488 received|inaccessible|unknown</btp:status>
3489 <btp:replyresponse-requested>true|false</btp:replyresponse-
3490 requested>
3491 <btp:qualifiers> ?
3492 ...qualifiers...
3493 </btp:qualifiers>
3494 </btp:superior-state>

INFERIOR_STATE

3495
3496
3497
3498


```
3499 <btpt:inferior-state id?>
3500   <btpt:target-additional-information> ?
3501     ...additional address information...
3502   </btpt:target-additional-information>
3503   <btpt:superior-identifier>...URI...</btpt:superior-identifier>
3504
3505   <btpt:inferior-identifier>...URI...</btpt:inferior-identifier>
3506   <btpt:status>active|inaccessible|unknown</btpt:status>
3507   <btpt:replyresponse-requested>true|false</btpt:replyresponse-
3508 requested>
3509   <btpt:qualifiers> ?
3510     ...qualifiers...
3511   </btpt:qualifiers>
3512 </btpt:inferior-state>
```

REDIRECT

```
3517 <btpt:redirect id?>
3518   <btpt:target-additional-information> ?
3519     ...additional address information...
3520   </btpt:target-additional-information>
3521   <btpt:superior-identifier>...URI...</btpt:superior-identifier> ?
3522   <btpt:inferior-identifier>...URI...</btpt:inferior-identifier>
3523   <btpt:old-address> +
3524     ...address...
3525   </btpt:old-address>
3526   <btpt:new-address> +
3527     ...address...
3528   </btpt:new-address>
3529   <btpt:qualifiers> ?
3530     ...qualifiers...
3531   </btpt:qualifiers>
3532 </btpt:redirect>
```

BEGIN

```
3536 <btpt:begin id?>
3537   <btpt:target-additional-information> ?
3538     ...additional address information...
3539   </btpt:target-additional-information>
3540   <btpt:reply-address> ?
3541     ...address...
3542   </btpt:reply-address>
3543   <btpt:transaction-type>cohesion|atom</btpt:transaction-type>
3544   <btpt:qualifiers> ?
3545     ...qualifiers...
3546   </btpt:qualifiers>
3547 </btpt:begin>
3548
3549
```

3550

BEGUN

3551

3552

```
<btp:begin id?>
  <btp:target-additional-information> ?
  ...additional address information...
</btp:target-additional-information>
<btp:decider-address> *
  ...address...
</btp:decider-address>
<btp:inferior-address> *
  ...address...
</btp:inferior-address>
<btp:transaction-identifier>...URI...</btp:transaction-
identifier>
<btp:qualifiers> ?
  ...qualifiers...
</btp:qualifiers>
</btp:begin>
```

3558

3559

3560

3561

PREPARE_INFERIORS

3571

3572

```
<btp:prepare-inferiors id?>
  <btp:target-additional-information> ?
  ...additional address information...
</btp:target-additional-information>
<btp:reply-address> ?
  ...address...
</btp:reply-address>
<btp:transaction-identifier>...URI...</btp:transaction-
identifier>
<btp:inferiors-list> ?
  <btp:inferior-handle>...URI...</btp:inferior-handle> +
</btp:inferiors-list>
<btp:qualifiers> ?
  ...qualifiers...
</btp:qualifiers>
</btp:prepare-inferiors>
```

3573

3574

3575

3576

3577

3578

3579

3580

3581

3582

3583

3584

3585

3586

3587

3588

3589

CONFIRM_TRANSACTION

3591

3592

```
<btp:confirm-transaction id?>
  <btp:target-additional-information> ?
  ...additional address information...
</btp:target-additional-information>
<btp:reply-address> ?
  ...address...
</btp:reply-address>
<btp:transaction-identifier>...URI...</btp:transaction-
identifier>
<btp:inferiors-list> ?
```

3593

3594

3595

3596

3597

3598

3599

3600

3601

```
3602     <btp:inferior-handle>...URI...</btp:inferior-handle> +
3603 </btp:inferiors-list>
3604 <btp:report-hazard>true|false</btp:report-hazard>
3605 <btp:qualifiers> ?
3606     ...qualifiers...
3607 </btp:qualifiers>
3608 </btp:confirm_transaction>
```

3609

3610

TRANSACTION_CONFIRMED

3611

3612

3613

```
<btp:transaction-confirmed id?>
  <btp:target-additional-information> ?
  ...additional address information...
</btp:target-additional-information>
```

3616

3617

3618

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

3619

3620

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

3621

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

3622

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

3623

```
</btp:transaction-confirmed>
```

3624

3625

CANCEL_TRANSACTION

3626

3627

3628

```
<btp:cancel-transaction id?>
  <btp:target-additional-information> ?
  ...additional address information...
</btp:target-additional-information>
```

3631

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

3632

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

3633

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

3634

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

3635

```
  <btp:report-hazard>true|false</btp:report-hazard>
```

3636

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

3637

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

3638

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

3639

```
</btp:cancel-transaction>
```

3640

3641

3642

CANCEL_INFERIORS

3643

3644

3645

```
<btp:cancel-inferiors id?>
  <btp:target-additional-information> ?
  ...additional address information...
</btp:target-additional-information>
```

3648

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

3649

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

3650

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

3651

```
3652     <btpt:transaction-identifier>...URI...</btpt:transaction-
3653 identifier> ?
3654     <btpt:inferiors-list>
3655         <btpt:inferior-handle>...URI...</btpt:inferior-handle> +
3656     </btpt:inferiors-list>
3657     <btpt:qualifiers> ?
3658         ...qualifiers...
3659     </btpt:qualifiers>
3660 </btpt:cancel-inferiors>
```

3661

3662

3663 TRANSACTION_CANCELLED

3664

```
3665     <btpt:transaction-cancelled id?>
3666         <btpt:target-additional-information> ?
3667             ...additional address information...
3668         </btpt:target-additional-information>
3669
3670         <btpt:transaction-identifier>...URI...</btpt:transaction-
3671 identifier>
3672         <btpt:qualifiers> ?
3673             ...qualifiers...
3674         </btpt:qualifiers>
3675     </btpt:transaction-cancelled>
```

3676

3677

3678 REQUEST_INFERIOR_STATUSES

3679

```
3680     <btpt:request-inferior-statuses id?>
3681         <btpt:target-additional-information> ?
3682             ...additional address information...
3683         </btpt:target-additional-information>
3684         <btpt:reply-address> ?
3685             ...address...
3686         </btpt:reply-address>
3687         <btpt:target-identifier>...URI...</btpt:target-identifier>
3688         <btpt:inferiors-list> ?
3689             <btpt:inferior-handle>...URI...</btpt:inferior-handle> +
3690         </btpt:inferiors-list>
3691         <btpt:qualifiers> ?
3692             ...qualifiers...
3693         </btpt:qualifiers>
3694     </btpt:request-inferior-statuses>
```

3695

3696

3697 INFERIOR_STATUSES

3698

```
3699     <btpt:inferior-statuses id?>
3700         <btpt:target-additional-information> ?
3701             ...additional address information...
3702         </btpt:target-additional-information>
```

3702

```

3703 <btpr:responders-identifier>...URI...</btpr:responders-identifier>
3704 <btpr:status-list>
3705   <btpr:status-item> +
3706     <btpr:inferior-handle>...URI...</btpr:inferior-handle>
3707     <btpr:status>active|resigned|preparing|prepared|
3708       autonomously-confirmed|autonomously-cancelled|
3709       confirming|confirmed|cancelling|cancelled|
3710       cancel-contradiction|confirm-contradiction|
3711       hazard|invalid</btpr:status>
3712     <btpr:qualifiers> ?
3713       ...qualifiers...
3714   </btpr:status-item>
3715 </btpr:status-list>
3716 <btpr:qualifiers> ?
3717   ...qualifiers...
3718 </btpr:qualifiers>
3719 </btpr:inferior-statuses>
3720
3721
3722

```

3723 **Standard qualifiers**

3724 The informal syntax for these messages assumes the namespace prefix “btpr” is associated
3725 with the URI “urn:oasis:names:tc:BTP:qualifiers”.

3726 **Transaction timelimit**

```

3727
3728
3729 <btprq:transaction-timelimit>
3730   <btprq:timelimit>
3731     ...time in seconds...
3732   </btprq:timelimit>
3733 </btprq:transaction-timelimit>
3734

```

3735 **Inferior timeout**

```

3736 <btprq:inferior-timeout>
3737   <btprq:timeout>
3738     ...time in seconds...
3739   </btprq:timeout>
3740   <btprq:intended-decision>confirm|cancel</btprq:intended-decision>
3741 </btprq:inferior-timeout>
3742

```

3743 **Minimum inferior timeout**

```

3744 <btprq:minimum-inferior-timeout>
3745   <btprq:minimum-timeout>
3746     ...time in seconds...
3747   </btprq:minimum-timeout>
3748 </btprq:minimum-inferior-timeout>
3749

```

3750 **Inferior name**

```

3751 <btprq:inferior-name>
3752 <btprq:inferior-name>
3753   ...string...

```

```
3754     </btpq:inferior-name>
3755 </btpq:inferior-name>
3756
```

3757 **Compounding of Messages**

3758

3759 Relating BTP to one another, in a “group” is represented by containing them within the
3760 btp:related-group element, with the related messages as child elements. The processing for
3761 the group is defined in the section “Groups – combinations of related messages”. For example

3762

```
3763 <btp:related-group>
3764     <btp:context-reply>
3765         ...<completion-status>related</completion-status> ...
3766     </btp:context-reply>
3767     <btp:enrol>...</btp:enrol>
3768     <btp:prepared>...</btp:prepared>
3769 </btp:related-group>
```

3770

3771 If the rules for the group state that the “target-address” of the abstract message is omitted, the
3772 corresponding target-address-information element shall be absent in the message in the
3773 related-group. The carrier protocol binding specifies how a relation between application and
3774 BTP messages is represented.

3775

3776 Bundling (semantically insignificant combination) of BTP messages and related groups is
3777 indicated with the "btp:messages" element, with the bundled messages and related groups as
3778 child elements. For example (confirming one and cancelling another inferiors of a cohesion):

3779

```
3780 <btp:messages>
3781     <btp:confirm>...</btp:confirm>
3782     <btp:cancel>...</btp:cancel>
3783 </btp:messages>
```

3784

3785

3786

3786

XML Schemas

3787

3788

3789

XML schema for BTP messages

3790

3791

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

3792

```
<schema
```

3793

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

3794

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

3795

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

3796

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

3797

3798

3799

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

3800

3801

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

3802

```
    <simpleContent>
```

3803

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

3804

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

3805

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

3806

```
      </extension>
```

3807

```
    </simpleContent>
```

3808

```
  </complexType>
```

3809

3810

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

3811

3812

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

3813

```
    <complexType>
```

3814

```
      <sequence>
```

3815

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

3816

```
      </sequence>
```

3817

```
    </complexType>
```

3818

```
  </element>
```

3819

3820

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

3821

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

3822

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

3823

```
  -->
```

3824

3825

3826

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

3827

3828

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

3829

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

3830

```
  </simpleType>
```

3831

3832

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

3833

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

3834

```
  </simpleType>
```

3835

3836

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

3837

```
    <sequence>
```

```

3838         <element name="binding-name" type="anyURI"/>
3839         <element name="binding-address" type="string"/>
3840         <element name="additional-information" type="btp:additional-
3841 information" minOccurs="0" />
3842     </sequence>
3843 </complexType>
3844
3845     <simpleType name="superior-type">
3846         <restriction base="string">
3847             <enumeration value="cohesion"/>
3848             <enumeration value="atom"/>
3849         </restriction>
3850     </simpleType>
3851
3852     <simpleType name="transaction-type">
3853         <restriction base="string">
3854             <enumeration value="cohesion"/>
3855             <enumeration value="atom"/>
3856         </restriction>
3857     </simpleType>
3858
3859     <!-- Compounding -->
3860
3861     <element name="messages">
3862         <complexType>
3863             <sequence>
3864                 <element ref="btp:message" minOccurs="0"
3865 maxOccurs="unbounded"/>
3866             </sequence>
3867         </complexType>
3868     </element>
3869
3870     <element name="related-group" substitutionGroup="btp:message">
3871         <complexType>
3872             <sequence>
3873                 <element ref="btp:message" minOccurs="0"
3874 maxOccurs="unbounded"/>
3875             </sequence>
3876         </complexType>
3877     </element>
3878
3879     <!-- Message set -->
3880
3881     <element name="message" abstract="true" />
3882
3883     <element name="context" substitutionGroup="btp:message">
3884         <complexType>
3885             <sequence>
3886                 <element name="superior-address" type="btp:address"
3887 maxOccurs="unbounded"/>
3888                 <element name="superior-identifier" type="btp:identifier"/>
3889
3890

```



```

3891         <element name="reply-address" type="btp:address"
3892 minOccurs="0"/>
3893         <element name="superior-type" type="btp:superior-type"/>
3894         <element ref="btp:qualifiers" minOccurs="0"/>
3895     </sequence>
3896     <attribute name="id" type="ID" use="optional"/>
3897 </complexType>
3898 </element>
3899
3900 <element name="context-reply" substitutionGroup="btp:message">
3901 <complexType>
3902 <sequence>
3903     <element name="target-additional-information"
3904 type="btp:additional-information" minOccurs="0"/>
3905     <element name="superior-identifier" type="btp:identifier"/>
3906     <element name="completion-status">
3907 <simpleType>
3908 <restriction base="string">
3909 <enumeration value="completed"/>
3910 <enumeration value="related"/>
3911 <enumeration value="repudiated"/>
3912 </restriction>
3913 </simpleType>
3914 </element>
3915     <element ref="btp:qualifiers" minOccurs="0"/>
3916 </sequence>
3917 <attribute name="id" type="ID"/>
3918 </complexType>
3919 </element>
3920
3921 <element name="request-status" substitutionGroup="btp:message">
3922 <complexType>
3923 <sequence>
3924     <element name="target-additional-information"
3925 type="btp:additional-information" minOccurs="0"/>
3926     <element name="reply-address" type="btp:address"
3927 minOccurs="0"/>
3928     <element name="target-identifier" type="btp:identifier"/>
3929     <element ref="btp:qualifiers" minOccurs="0"/>
3930 </sequence>
3931 <attribute name="id" type="ID"/>
3932 </complexType>
3933 </element>
3934
3935 <element name="status" substitutionGroup="btp:message">
3936 <complexType>
3937 <sequence>
3938     <element name="target-additional-information"
3939 type="btp:additional-information" minOccurs="0"/>
3940     <element name="responders-identifier"
3941 type="btp:identifier"/>
3942     <element name="status-value">
3943 <simpleType>

```

```

3944         <restriction base="string">
3945             <enumeration value="created"/>
3946             <enumeration value="enrolling"/>
3947             <enumeration value="active"/>
3948             <enumeration value="resigning"/>
3949             <enumeration value="resigned"/>
3950             <enumeration value="preparing"/>
3951             <enumeration value="prepared"/>
3952             <enumeration value="confirming"/>
3953             <enumeration value="confirmed"/>
3954             <enumeration value="cancelling"/>
3955             <enumeration value="cancelled"/>
3956             <enumeration value="cancel-contradiction"/>
3957             <enumeration value="confirm-contradiction"/>
3958             <enumeration value="hazard"/>
3959             <enumeration value="contradicted"/>
3960             <enumeration value="unknown"/>
3961             <enumeration value="inaccessible"/>
3962         </restriction>
3963     </simpleType>
3964 </element>
3965     <element ref="btp:qualifiers" minOccurs="0"/>
3966 </sequence>
3967     <attribute name="id" type="ID"/>
3968 </complexType>
3969 </element>
3970
3971 <element name="fault" substitutionGroup="btp:message">
3972     <complexType>
3973         <sequence>
3974             <element name="target-additional-information"
3975 type="btp:additional-information" minOccurs="0"/>
3976             <element name="superior-identifier" type="btp:identifier"
3977 minOccurs="0"/>
3978             <element name="inferior-identifier" type="btp:identifier"
3979 minOccurs="0"/>
3980             <element name="fault-type">
3981                 <simpleType>
3982                     <restriction base="string">
3983                         <enumeration value="communication-failure"/>
3984                         <enumeration value="duplicate-inferior"/>
3985                         <enumeration value="general"/>
3986                         <enumeration value="invalid-decider"/>
3987                         <enumeration value="invalid-inferior"/>
3988                         <enumeration value="invalid-superior"/>
3989                         <enumeration value="status-refused"/>
3990                         <enumeration value="invalid-terminator"/>
3991                         <enumeration value="unknown-parameter"/>
3992                         <enumeration value="unknown-transaction"/>
3993                         <enumeration value="unsupported-qualifier"/>
3994                         <enumeration value="wrong-state"/>
3995                     </restriction>
3996                 </simpleType>

```

```

3997         </element>
3998         <element name="fault-data" type="anyType" minOccurs="0"/>
3999         <element ref="btp:qualifiers" minOccurs="0"/>
4000     </sequence>
4001     <attribute name="id" type="ID"/>
4002 </complexType>
4003 </element>
4004
4005 <element name="enrol" substitutionGroup="btp:message">
4006     <complexType>
4007         <sequence>
4008             <element name="target-additional-information"
4009 type="btp:additional-information" minOccurs="0"/>
4010             <element name="superior-identifier" type="btp:identifier"/>
4011             <element name="replyresponse-requested" type="boolean"/> |
4012             <element name="reply-address" type="btp:address"
4013 minOccurs="0"/>
4014             <element name="inferior-address" type="btp:address"
4015 minOccurs="1" maxOccurs="unbounded"/>
4016             <element name="inferior-identifier" type="btp:identifier"/>
4017             <element ref="btp:qualifiers" minOccurs="0"/>
4018         </sequence>
4019         <attribute name="id" type="ID"/>
4020     </complexType>
4021 </element>
4022
4023
4024 <element name="enrolled" substitutionGroup="btp:message">
4025     <complexType>
4026         <sequence>
4027             <element name="target-additional-information"
4028 type="btp:additional-information" minOccurs="0"/>
4029             <element name="inferior-identifier" type="btp:identifier"/>
4030             <element ref="btp:qualifiers" minOccurs="0"/>
4031         </sequence>
4032         <attribute name="id" type="ID"/>
4033     </complexType>
4034 </element>
4035
4036 <element name="resign" substitutionGroup="btp:message">
4037     <complexType>
4038         <sequence>
4039             <element name="target-additional-information"
4040 type="btp:additional-information" minOccurs="0"/>
4041             <element name="superior-identifier" type="btp:identifier"/>
4042             <element name="inferior-identifier" type="btp:identifier"/>
4043             <element name="response-requested" type="boolean"/>
4044             <element ref="btp:qualifiers" minOccurs="0"/>
4045         </sequence>
4046         <attribute name="id" type="ID"/>
4047     </complexType>
4048 </element>
4049

```

```

4050     <element name="resigned" substitutionGroup="btp:message">
4051         <complexType>
4052             <sequence>
4053                 <element name="target-additional-information"
4054 type="btp:additional-information" minOccurs="0"/>
4055                 <element name="inferior-identifier" type="btp:identifier"/>
4056                 <element ref="btp:qualifiers" minOccurs="0"/>
4057             </sequence>
4058             <attribute name="id" type="ID"/>
4059         </complexType>
4060     </element>
4061
4062     <element name="prepare" substitutionGroup="btp:message">
4063         <complexType>
4064             <sequence>
4065                 <element name="target-additional-information"
4066 type="btp:additional-information" minOccurs="0"/>
4067                 <element name="inferior-identifier" type="btp:identifier"/>
4068                 <element ref="btp:qualifiers" minOccurs="0"/>
4069             </sequence>
4070             <attribute name="id" type="ID"/>
4071         </complexType>
4072     </element>
4073
4074     <element name="prepared" substitutionGroup="btp:message">
4075         <complexType>
4076             <sequence>
4077                 <element name="target-additional-information"
4078 type="btp:additional-information" minOccurs="0"/>
4079                 <element name="superior-identifier" type="btp:identifier"/>
4080                 <element name="inferior-identifier" type="btp:identifier"/>
4081                 <element name="default-is-cancel" type="boolean"/>
4082                 <element ref="btp:qualifiers" minOccurs="0"/>
4083             </sequence>
4084             <attribute name="id" type="ID"/>
4085         </complexType>
4086     </element>
4087
4088     <element name="confirm" substitutionGroup="btp:message">
4089         <complexType>
4090             <sequence>
4091                 <element name="target-additional-information"
4092 type="btp:additional-information" minOccurs="0"/>
4093                 <element name="inferior-identifier" type="btp:identifier"/>
4094                 <element ref="btp:qualifiers" minOccurs="0"/>
4095             </sequence>
4096             <attribute name="id" type="ID"/>
4097         </complexType>
4098     </element>
4099
4100     <element name="confirmed" substitutionGroup="btp:message">
4101         <complexType>
4102             <sequence>

```

```

4103         <element name="target-additional-information"
4104 type="btp:additional-information" minOccurs="0"/>
4105         <element name="superior-identifier" type="btp:identifier"/>
4106         <element name="inferior-identifier" type="btp:identifier"/>
4107         <element name="confirmed-received" type="boolean"/>
4108         <element ref="btp:qualifiers" minOccurs="0"/>
4109     </sequence>
4110     <attribute name="id" type="ID"/>
4111 </complexType>
4112 </element>
4113
4114     <element name="cancel" substitutionGroup="btp:message">
4115         <complexType>
4116             <sequence>
4117                 <element name="target-additional-information"
4118 type="btp:additional-information" minOccurs="0"/>
4119                 <element name="inferior-identifier" type="btp:identifier"/>
4120                 <element name="reply-address" type="btp:address"
4121 minOccurs="0"/>
4122                 <element ref="btp:qualifiers" minOccurs="0"/>
4123             </sequence>
4124             <attribute name="id" type="ID"/>
4125         </complexType>
4126     </element>
4127
4128     <element name="cancelled" substitutionGroup="btp:message">
4129         <complexType>
4130             <sequence>
4131                 <element name="target-additional-information"
4132 type="btp:additional-information" minOccurs="0"/>
4133                 <element name="superior-identifier" type="btp:identifier"/>
4134                 <element name="inferior-identifier" type="btp:identifier"
4135 minOccurs="0"/>
4136                 <element ref="btp:qualifiers" minOccurs="0"/>
4137             </sequence>
4138             <attribute name="id" type="ID"/>
4139         </complexType>
4140     </element>
4141
4142     <element name="confirm-one-phase" substitutionGroup="btp:message">
4143         <complexType>
4144             <sequence>
4145                 <element name="target-additional-information"
4146 type="btp:additional-information" minOccurs="0"/>
4147                 <element name="inferior-identifier" type="btp:identifier"/>
4148                 <element name="report-hazard" type="boolean"/>
4149                 <element ref="btp:qualifiers" minOccurs="0"/>
4150             </sequence>
4151             <attribute name="id" type="ID"/>
4152         </complexType>
4153     </element>
4154
4155     <element name="hazard" substitutionGroup="btp:message">

```

```

4156     <complexType>
4157         <sequence>
4158             <element name="target-additional-information"
4159 type="btp:additional-information" minOccurs="0"/>
4160             <element name="superior-identifier" type="btp:identifier"/>
4161             <element name="inferior-identifier" type="btp:identifier"/>
4162             <element name="level">
4163                 <simpleType>
4164                     <restriction base="string">
4165                         <enumeration value="mixed"/>
4166                         <enumeration value="possible"/>
4167                     </restriction>
4168                 </simpleType>
4169             </element>
4170             <element ref="btp:qualifiers" minOccurs="0"/>
4171         </sequence>
4172         <attribute name="id" type="ID"/>
4173     </complexType>
4174 </element>
4175
4176     <element name="contradiction" substitutionGroup="btp:message">
4177         <complexType>
4178             <sequence>
4179                 <element name="target-additional-information"
4180 type="btp:additional-information" minOccurs="0"/>
4181                 <element name="inferior-identifier" type="btp:identifier"/>
4182                 <element ref="btp:qualifiers" minOccurs="0"/>
4183             </sequence>
4184             <attribute name="id" type="ID"/>
4185         </complexType>
4186 </element>
4187
4188     <element name="superior-state" substitutionGroup="btp:message">
4189         <complexType>
4190             <sequence>
4191                 <element name="target-additional-information"
4192 type="btp:additional-information" minOccurs="0"/>
4193                 <element name="inferior-identifier" type="btp:identifier"/>
4194                 <element name="status">
4195                     <simpleType>
4196                         <restriction base="string">
4197                             <enumeration value="active"/>
4198                             <enumeration value="prepared-received"/>
4199                             <enumeration value="inaccessible"/>
4200                             <enumeration value="unknown"/>
4201                         </restriction>
4202                     </simpleType>
4203                 </element>
4204                 <element name="replyresponse-requested" type="boolean"/>
4205                 <element ref="btp:qualifiers" minOccurs="0"/>
4206             </sequence>
4207             <attribute name="id" type="ID"/>
4208         </complexType>

```

```

4209     </element>
4210
4211     <element name="inferior-state" substitutionGroup="btp:message">
4212         <complexType>
4213             <sequence>
4214                 <element name="target-additional-information"
4215 type="btp:additional-information" minOccurs="0"/>
4216                 <element name="superior-identifier" type="btp:identifier"/>
4217                 <element name="inferior-identifier" type="btp:identifier"/>
4218                 <element name="status">
4219                     <simpleType>
4220                         <restriction base="string">
4221                             <enumeration value="active"/>
4222                             <enumeration value="inaccessible"/>
4223                             <enumeration value="unknown"/>
4224                         </restriction>
4225                     </simpleType>
4226                 </element>
4227                 <element name="replyresponse-requested" type="boolean"/>
4228                 <element ref="btp:qualifiers" minOccurs="0"/>
4229             </sequence>
4230             <attribute name="id" type="ID"/>
4231         </complexType>
4232     </element>
4233
4234     <element name="redirect" substitutionGroup="btp:message">
4235         <complexType>
4236             <sequence>
4237                 <element name="target-additional-information"
4238 type="btp:additional-information" minOccurs="0"/>
4239                 <element name="superior-identifier" type="btp:identifier"
4240 minOccurs="0"/>
4241                 <element name="inferior-identifier" type="btp:identifier"
4242 />
4243                 <element name="old-address" type="btp:address"
4244 maxOccurs="unbounded"/>
4245                 <element name="new-address" type="btp:address"
4246 maxOccurs="unbounded"/>
4247                 <element ref="btp:qualifiers" minOccurs="0"/>
4248             </sequence>
4249             <attribute name="id" type="ID"/>
4250         </complexType>
4251     </element>
4252
4253
4254     <element name="begin" substitutionGroup="btp:message">
4255         <complexType>
4256             <sequence>
4257                 <element name="target-additional-information"
4258 type="btp:additional-information" minOccurs="0"/>
4259                 <element name="reply-address" type="btp:address"
4260 minOccurs="0"/>
4261                 <element name="transaction-type" type="btp:superior-type"/>

```

```

4262         <element ref="btp:qualifiers" minOccurs="0"/>
4263     </sequence>
4264     <attribute name="id" type="ID"/>
4265 </complexType>
4266 </element>
4267
4268     <element name="begun" substitutionGroup="btp:message">
4269         <complexType>
4270             <sequence>
4271                 <element name="target-additional-information"
4272 type="btp:additional-information" minOccurs="0"/>
4273                 <element name="decider-address" type="btp:address"
4274 minOccurs="0" maxOccurs="unbounded"/>
4275                 <element name="transaction-identifier"
4276 type="btp:identifier" minOccurs="0"/>
4277                 <element name="inferior-handle" type="btp:identifier"
4278 minOccurs="0"/>
4279                 <element name="inferior-address" type="btp:address"
4280 minOccurs="0" maxOccurs="unbounded"/>
4281                 <element ref="btp:qualifiers" minOccurs="0"/>
4282             </sequence>
4283             <attribute name="id" type="ID"/>
4284         </complexType>
4285     </element>
4286
4287     <element name="prepare-inferiors" substitutionGroup="btp:message">
4288         <complexType>
4289             <sequence>
4290                 <element name="target-additional-information"
4291 type="btp:additional-information" minOccurs="0"/>
4292                 <element name="reply-address" type="btp:address"
4293 minOccurs="0"/>
4294                 <element name="transaction-identifier"
4295 type="btp:identifier"/>
4296                 <element name="inferiors-list" minOccurs="0">
4297                     <complexType>
4298                         <sequence>
4299                             <element name="inferior-handle"
4300 type="btp:identifier" maxOccurs="unbounded"/>
4301                         </sequence>
4302                     </complexType>
4303                 </element>
4304                 <element ref="btp:qualifiers" minOccurs="0"/>
4305             </sequence>
4306             <attribute name="id" type="ID"/>
4307         </complexType>
4308     </element>
4309
4310     <element name="confirm-transaction" substitutionGroup="btp:message">
4311         <complexType>
4312             <sequence>
4313                 <element name="target-additional-information"
4314 type="btp:additional-information" minOccurs="0"/>

```



```

4315         <element name="reply-address" type="btp:address"
4316 minOccurs="0"/>
4317         <element name="transaction-identifier"
4318 type="btp:identifier"/>
4319         <element name="inferiors-list" minOccurs="0">
4320             <complexType>
4321                 <sequence>
4322                     <element name="inferior-handle"
4323 type="btp:identifier" maxOccurs="unbounded"/>
4324                 </sequence>
4325             </complexType>
4326         </element>
4327         <element name="report-hazard" type="boolean"/>
4328         <element ref="btp:qualifiers" minOccurs="0"/>
4329     </sequence>
4330     <attribute name="id" type="ID"/>
4331 </complexType>
4332 </element>
4333
4334     <element name="transaction-confirmed" substitutionGroup="btp:message">
4335         <complexType>
4336             <sequence>
4337                 <element name="target-additional-information"
4338 type="btp:additional-information" minOccurs="0"/>
4339                 <element name="transaction-identifier"
4340 type="btp:identifier"/>
4341                 <element ref="btp:qualifiers" minOccurs="0"/>
4342             </sequence>
4343             <attribute name="id" type="ID"/>
4344         </complexType>
4345     </element>
4346
4347     <element name="cancel-transaction" substitutionGroup="btp:message">
4348         <complexType>
4349             <sequence>
4350                 <element name="target-additional-information"
4351 type="btp:additional-information" minOccurs="0"/>
4352                 <element name="reply-address" type="btp:address"
4353 minOccurs="0"/>
4354                 <element name="transaction-identifier"
4355 type="btp:identifier"/>
4356                 <element name="report-hazard" type="boolean"/>
4357                 <element ref="btp:qualifiers" minOccurs="0"/>
4358             </sequence>
4359             <attribute name="id" type="ID"/>
4360         </complexType>
4361     </element>
4362
4363     <element name="cancel-inferiors" substitutionGroup="btp:message">
4364         <complexType>
4365             <sequence>
4366                 <element name="target-additional-information"
4367 type="btp:additional-information" minOccurs="0"/>

```

```

4368         <element name="reply-address" type="btp:address"
4369 minOccurs="0"/>
4370         <element name="transaction-identifier"
4371 type="btp:identifier" minOccurs="0"/>
4372         <element name="inferiors-list">
4373             <complexType>
4374                 <sequence>
4375                     <element name="inferior-handle"
4376 type="btp:identifier" maxOccurs="unbounded"/>
4377                 </sequence>
4378             </complexType>
4379         </element>
4380         <element ref="btp:qualifiers" minOccurs="0"/>
4381     </sequence>
4382     <attribute name="id" type="ID"/>
4383 </complexType>
4384 </element>
4385
4386     <element name="transaction-cancelled" substitutionGroup="btp:message">
4387         <complexType>
4388             <sequence>
4389                 <element name="target-additional-information"
4390 type="btp:additional-information" minOccurs="0"/>
4391                 <element name="transaction-identifier"
4392 type="btp:identifier"/>
4393                 <element ref="btp:qualifiers" minOccurs="0"/>
4394             </sequence>
4395             <attribute name="id" type="ID"/>
4396         </complexType>
4397     </element>
4398
4399     <element name="request-inferior-statuses"
4400 substitutionGroup="btp:message">
4401         <complexType>
4402             <sequence>
4403                 <element name="target-additional-information"
4404 type="btp:additional-information" minOccurs="0"/>
4405                 <element name="reply-address" type="btp:address"
4406 minOccurs="0"/>
4407                 <element name="target-identifier" type="btp:identifier"/>
4408                 <element name="inferiors-list" minOccurs="0">
4409                     <complexType>
4410                         <sequence>
4411                             <element name="inferior-handle"
4412 type="btp:identifier" maxOccurs="unbounded"/>
4413                         </sequence>
4414                     </complexType>
4415                 </element>
4416                 <element ref="btp:qualifiers" minOccurs="0"/>
4417             </sequence>
4418             <attribute name="id" type="ID"/>
4419         </complexType>
4420     </element>

```

```

4421
4422     <element name="inferior-statuses" substitutionGroup="btp:message">
4423         <complexType>
4424             <sequence>
4425                 <element name="target-additional-information"
4426 type="btp:additional-information" minOccurs="0"/>
4427                 <element name="responders-identifier"
4428 type="btp:identifier"/>
4429                 <element name="status-list">
4430                     <complexType>
4431                         <sequence>
4432                             <element name="status-item" maxOccurs="unbounded">
4433                                 <complexType>
4434                                     <sequence>
4435                                         <element name="inferior-handle"
4436 type="btp:identifier"/>
4437                                         <element name="status">
4438                                             <simpleType>
4439                                                 <restriction base="string">
4440                                                     <enumeration value="active"/>
4441                                                     <enumeration value="resigned"/>
4442                                                     <enumeration value="preparing"/>
4443                                                     <enumeration value="prepared"/>
4444                                                     <enumeration value="autonomously-confirmed"/>
4445                                                     <enumeration value="autonomously-cancelled"/>
4446                                                     <enumeration value="confirming"/>
4447                                                     <enumeration value="confirmed"/>
4448                                                     <enumeration value="cancelling"/>
4449                                                     <enumeration value="cancelled"/>
4450                                                     <enumeration value="cancel-contradiction"/>
4451                                                     <enumeration value="confirm-contradiction"/>
4452                                                     <enumeration value="hazard"/>
4453                                                     <enumeration value="invalid"/>
4454                                                 </restriction>
4455                                             </simpleType>
4456                                         </element>
4457                                         <element ref="btp:qualifiers" minOccurs="0"/>
4458                                     </sequence>
4459                                 </complexType>
4460                             </element>
4461                         </sequence>
4462                     </complexType>
4463                 </element>
4464                 <element ref="btp:qualifiers" minOccurs="0"/>
4465             </sequence>
4466             <attribute name="id" type="ID"/>
4467         </complexType>
4468     </element>
4469
4470
4471 </schema>
4472

```

XML schema for standard qualifiers

4473
4474
4475
4476
4477
4478
4479
4480
4481
4482
4483
4484
4485
4486
4487
4488
4489
4490
4491
4492
4493
4494
4495
4496
4497
4498
4499
4500
4501
4502
4503
4504
4505
4506
4507
4508
4509
4510
4511
4512
4513
4514
4515
4516
4517
4518
4519
4520
4521
4522
4523
4524

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

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

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

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

```
4525         <element name="minimum-timeout"
4526 type="nonNegativeInteger" />
4527         </sequence>
4528     </extension>
4529 </complexContent>
4530 </complexType>
4531 </element>
4532
4533 <element name="inferior-name" substitutionGroup="btp:qualifier">
4534     <complexType>
4535         <complexContent>
4536             <extension base="btp:qualifier-type">
4537                 <sequence>
4538                     <element name="inferior-name" type="string" />
4539                 </sequence>
4540             </extension>
4541         </complexContent>
4542     </complexType>
4543 </element>
4544
4545 </schema>
4546
```

4546
4547

4548 **Carrier Protocol Bindings**

4549
4550
4551
4552
4553
4554
4555
4556
4557
4558
4559

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.

4560 **Carrier Protocol Binding Proforma**

4561
4562
4563

A BTP carrier binding specification should provide the following information:

4564
4565
4566
4567
4568
4569
4570

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

4571
4572
4573
4574

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

4575
4576
4577
4578

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.

4579
4580
4581
4582
4583
4584
4585
4586
4587

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.

4588
4589
4590
4591

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

4592 “the CONTEXT may be carried as a parameter of an application invocation”). Alternatively,
4593 the binding may specify a general method that represents the relationship between application
4594 and BTP messages.

4595
4596 **Implicit messages:** This section specifies which BTP messages, if any, are not sent explicitly
4597 but are treated as implicit in application messages or other BTP messages. This may depend
4598 on particular parameter values of the BTP messages or the application messages.

4599
4600 **Faults:** The relationship between the fault and exception reporting mechanisms of the carrier
4601 protocol and of BTP shall be defined. This may include definition of which carrier protocol
4602 exceptions are equivalent to a FAULT/communication-failure message.

4603
4604 **Relationship to other bindings:** Any relationship to other bindings is defined in this section.
4605 If BTP addresses with different bindings are be considered to match (for purposes of
4606 identifying the peer Superior/Inferior and redirection), this should be specified here.

4607
4608 **Limitations on BTP use:** Any limitations on the full range of BTP functionality that are
4609 imposed by use of this binding should be listed. This would include limitations on which
4610 messages can be sent, which event sequences are supported and restrictions on parameter
4611 values. Such limitations may reduce the usefulness of an implementation, but may be
4612 appropriate in certain environments.

4613
4614 **Other:** Other features of the binding, especially any that will potentially affect interoperation
4615 should be specified here. This may include restrictions or requirements on the use or support
4616 of optional carrier parameters or mechanisms.

4617

4618 **Bindings for request/response carrier protocols**

4619

4620 BTP does not generally follow request/response pattern. In particular, on the outcome
4621 relationship either side may initiate a message – this is an essential part of the presume-abort
4622 recovery paradigm although it is not limited to recovery cases. However, there are some BTP
4623 messages, especially in the control relationship, that do have a request/response pattern.
4624 Many (potential) carrier protocols (e.g. HTTP) do have a request/response pattern. The
4625 specification of a binding specification to a request/response carrier protocol needs to state
4626 what rules apply – which messages can be carried by requests, which by responses. The
4627 simplest rule is to send all BTP messages on requests, and let the carrier responses travel back
4628 empty. This would be inefficient in use of network resources, and possibly inconvenient
4629 when used for the BTP request/response pairs.

4630

4631 This section defines a set of rules that allow more efficient use of the carrier, while allowing
4632 the initiator of a BTP request/response pair to ensure the BTP response is sent back on the
4633 carrier response. These rules are specified in this section to enable binding specifications to
4634 reference them, without requiring each binding specification to repeat similar information.

4635

4636 A binding to a request/response carrier is not required to use these rules. It may define other
4637 rules.

4638

4639
4640
4641
4642
4643
4644
4645
4646
4647
4648
4649
4650
4651
4652
4653
4654
4655
4656
4657
4658
4659
4660
4661
4662
4663
4664
4665
4666
4667
4668
4669
4670
4671
4672
4673
4674
4675
4676
4677
4678
4679
4680
4681
4682
4683
4684

Request/response exploitation rules

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

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

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

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

The rules:

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

- 4685 e) A BTP actor that receives a carrier protocol request carrying BTP messages that
4686 do have a “reply-address”, or which initiate processing that produces BTP
4687 messages whose target binding address matches the origin of the request, **may**
4688 freely choose whether to use the carrier protocol response for the replies, or to
4689 send back an “empty carrier protocol response”, and send the BTP replies in a
4690 separately initiated carrier protocol request. The characteristics of an “empty
4691 carrier protocol response” shall be stated in the particular binding specification.
4692
- 4693 f) A BTP actor that sends BTP messages on a carrier protocol request **must** be able
4694 to accept returning BTP messages on the corresponding carrier protocol response
4695 and, if the actor has offered an address on which it will receive carrier requests,
4696 must be able to accept “replying” BTP messages on a separate carrier protocol
4697 request.
4698

4699 SOAP Binding

4700 This binding describes how BTP messages will be carried using SOAP as in the [SOAP 1.1](#)
4701 specification, using the SOAP literal messaging style conventions. If no application message
4702 is sent at the same time, the BTP messages are contained within the SOAP Body element. If
4703 application messages are sent, the BTP messages are contained in the SOAP Header element.
4704

4705 **Binding name:** soap-http-1
4706

4707 **Binding address format:** shall be a URL, of type HTTP.
4708

4709 **BTP message representation:** The string representation of the XML, as specified in the
4710 XML schema defined in this document shall be used. The BTP XML messages are embedded
4711 in the SOAP message without the use of any specific encoding rules (literal style SOAP
4712 message); hence the encodingStyle attribute need not be set or can be set to an empty string.
4713

4714 **Mapping for BTP messages (unrelated):** The “request/response exploitation” rules shall be
4715 used.
4716

4717 BTP messages sent on an HTTP request or HTTP response which is not carrying an
4718 application message, the messages are contained in a single btp:messages element which is
4719 the immediate child element of the SOAP Body element.
4720

4721 An “empty carrier protocol response” sent after receiving an HTTP request containing a
4722 btp:messages element in the SOAP Body and the implementation BTP actor chooses just to
4723 reply at the lower level (and when the request/response exploitation rules allow an empty
4724 carrier protocol response), shall be any of:
4725

- 4726 a) an empty HTTP response
- 4727 b) an HTTP response containing an empty SOAP Envelope
- 4728 c) an HTTP response containing a SOAP Envelope containing a single, empty
4729 btp:messages element.
4730

4731 The receiver (the initial sender of the HTTP request) shall treat these in the same way – they
4732 have no effect on the BTP sequence (other than indicating that the earlier sending did not
4733 cause a communication failure.)
4734

4735
4736
4737 If an application message is being sent at the same time, the mapping for related messages
4738 shall be used, as if the BTP messages were related to the application message. (There is no
4739 ambiguity in whether the BTP messages are related, because only CONTEXT and ENROL
4740 can be related to an application message.)
4741

4742 **Mapping for BTP messages related to application messages:** All BTP messages sent with
4743 an application message, whether related to the application message or not, shall be sent in a
4744 single btp:messages element in the SOAP Header. There shall be precisely one btp:messages
4745 element in the SOAP Header.
4746

4747 The “request/response exploitation” rules shall apply to the BTP messages carried in the
4748 SOAP Header, as if they had been carried in a SOAP Body, unrelated to an application
4749 message, sent to the same binding address.

4750 Note – The application protocol itself (which is using the SOAP Body) may
4751 use the SOAP RPC or document approach – this is determined by the
4752 application.

4753 Only CONTEXT and ENROL messages are related (&) to application messages. If there is
4754 only one CONTEXT or one ENROL message present in the SOAP Header, it is assumed to
4755 be related to the whole of the application message in the SOAP Body. If there are multiple
4756 CONTEXT or ENROL messages, any relation of these BTP messages shall be indicated by
4757 application specific means.

4758 Note 1 – An application protocol could use references to the ID values of the
4759 BTP messages to indicate relation between BTP CONTEXT or ENROL
4760 messages and the application message.

4761 Note 2 -- However indicated, what the relatedness means, or even whether it
4762 has any significance at all, is a matter for the application.

4763
4764 **Implicit messages:** A SOAP FAULT, or other communication failure received in response to
4765 a SOAP request that had a CONTEXT in the SOAP Header shall be treated as if a
4766 CONTEXT_REPLY/repudiated had been received. See also the discussion under “other”
4767 about the SOAP mustUnderstand attribute.
4768

4769 **Faults:** A SOAP FAULT or other communication failure shall be treated as
4770 FAULT/communication-failure.
4771

4772 **Relationship to other bindings:** A BTP address for Superior or Inferior that has the binding
4773 string “soap-http-1” is considered to match one that has the binding string “soap-attachments-
4774 http-1” if the binding address and additional information fields match.

4775
4776 **Limitations on BTP use:** None

4777
4778 **Other:** The SOAP BTP binding does not make use of SOAPAction HTTP header or actor
4779 attribute. The SOAPAction HTTP header is left to be application specific when there are
4780 application messages in the SOAP Body, as an already existing web service that is being
4781 upgraded to use BTP might have already made use of SOAPAction. The SOAPAction HTTP
4782 header shall be omitted when the SOAP message carries only BTP messages in the SOAP
4783 Body.

4784
4785 The SOAP mustUnderstand attribute, when used on the btp:messages containing a BTP
4786 CONTEXT, ensures that the receiver (server, as a whole) supports BTP sufficiently to
4787 determine whether any enrolments are necessary and replies with CONTEXT_REPLY as
4788 appropriate. The sender of the CONTEXT (and related application message) can use this to
4789 ensure that the application work is performed as part of the business transaction, assuming the
4790 receiver’s SOAP implementation supports the mustUnderstand attribute. If mustUnderstand if
4791 false, a receiver can ignore the CONTEXT (if BTP is not supported there), and no
4792 CONTEXT_REPLY will be returned. It is a local option on the sender (client) side whether
4793 the absence of a CONTEXT_REPLY is assumed to be equivalent to aCONTEXT_REPLY/ok
4794 (and the business transaction allowed to proceed to confirmation).

4795
4796 Note – some SOAP implementations may not support the mustUnderstand attribute sufficiently to
4797 enforce these requirements.

4798 **Example scenario using SOAP binding**

4799
4800 The example below shows an application request with CONTEXT message sent from
4801 client.example.com (which includes the Superior) to services.example.com (Service).

```
4802  
4803  
4804 <soap:Envelope  
4805     xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"  
4806     soap:encodingStyle="">  
4807  
4808     <soap:Header>  
4809  
4810         <btp:messages xmlns:btp="urn:oasis:names:tc:BTP:xml">  
4811             <btp:context superior-type="atom">  
4812                 <btp:superior-address>  
4813                     <btp:binding>soap-http-1</btp:binding>  
4814                     <btp:binding-  
4815 address>http://client.example.com/soaphandler</btp:binding-  
4816 address>  
4817                     <btp:additional-information>btpengine</btp:additional-  
4818 information>  
4819                 </btp:superior-address>
```

```

4820         <btp:superior-
4821 identifier>http://example.com/1001</btp:superior-identifier>
4822         <btp:qualifiers>
4823             <btpq:transaction-timelimit
4824 xmlns:btpq="urn:oasis:names:tc:BTP:qualifiers"><btpq:timelimit>180
4825 0</btpq:timelimit></btpq:transaction-timelimit>
4826             </btp:qualifiers>
4827         </btp:context>
4828     </btp:messages>
4829
4830 </soap:Header>
4831
4832 <soap:Body>
4833
4834     <ns1:orderGoods
4835 xmlns:ns1="http://example.com/2001/Services/xyzgoods">
4836         <custID>ABC8329045</custID>
4837         <itemID>224352</itemID>
4838         <quantity>5</quantity>
4839     </ns1:orderGoods>
4840
4841 </soap:Body>
4842
4843 </soap:Envelope>
4844

```

4845
 4846 The example below shows CONTEXT_REPLY and a related ENROL message sent from
 4847 services.example.com to client.example.com, in reply to the previous message. There is no
 4848 application response, so the BTP messages are in the SOAP Body. The ENROL message
 4849 does not contain the target-additional-information, since the grouping rules for
 4850 CONTEXT_REPLY & ENROL omit the "target_address" (the receiver of this example
 4851 remembers the superior address from the original CONTEXT)

```

4852
4853 <soap:Envelope
4854     xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
4855     soap:encodingStyle="">
4856
4857     <soap:Header>
4858     </soap:Header>
4859
4860     <soap:Body>
4861
4862         <btp:messages xmlns:btp="urn:oasis:names:tc:BTP:xml">
4863             <btp:related-group>
4864                 <btp:context-reply>
4865                     <btp:target-additional-information>btpengine</btp:target-
4866 additional-information>
4867                     <btp:superior-
4868 identifier>http://example.com/1001</btp:superior-identifier>
4869                     <completion-status>related</completion-status>
4870                 </btp:context-reply>
4871

```

```

4872         <btpe:enrol replyresponse-requested="false">
4873             <btpe:target-additional-
4874 information>btpeengine</btpe:target-additional-information>
4875             <btpe:superior-
4876 identifier>http://example.com/1001</btpe:superior-identifier>
4877             <btpe:inferior-address>
4878                 <btpe:binding>soap-http-1</btpe:binding>
4879                 <btpe:binding-address>
4880                     http://services.example.com/soaphandler
4881                 </btpe:binding-address>
4882             </btpe:inferior-address>
4883             <btpe:inferior-identifier>
4884                 http://example.com/AAAB
4885             </btpe:inferior-identifier>
4886         </btpe:enrol>
4887     </btpe:related-group>
4888 </btpe:messages>
4889 </soap:Body>
4890 </soap:Envelope>

```

4897 SOAP + Attachments Binding

4898
4899 This binding describes how BTP messages will be carried using SOAP as in the [SOAP](#)
4900 [Messages with Attachments](#) specification. It is a superset of the Basic SOAP binding, soap-
4901 http-1. The two bindings only differ when application messages are sent.

4902
4903 **Binding name:** soap-attachments-http-1

4904
4905 **Binding address format:** as for soap-http-1

4906
4907 **BTP message representation:** As for soap-http-1

4908
4909 **Mapping for BTP messages (unrelated):** As for “soap-http-1”, except the SOAP Envelope
4910 containing the SOAP Body containing the BTP messages shall be in a MIME body part, as
4911 specified in [SOAP Messages with Attachments](#) specification. If an application message is
4912 being sent at the same time, the mapping for related messages for this binding shall be used,
4913 as if the BTP messages were related to the application message(s).

4914
4915 **Mapping for BTP messages related to application messages:** MIME packaging shall be
4916 used. One of the MIME multipart/related parts shall contain a SOAP Envelope, whose SOAP
4917 Headers element shall contain precisely one btpe:messages element, containing any BTP
4918 messages. Any BTP CONTEXT in the btpe:messages is considered to be related to the
4919 application message(s) in the SOAP Body, and to also any of the MIME parts referenced
4920 from the SOAP Body (using the “href” attribute).

4921
4922
4923
4924
4925
4926
4927
4928
4929
4930
4931
4932
4933
4934
4935
4936
4937
4938
4939
4940
4941
4942
4943
4944
4945
4946
4947
4948
4949
4950
4951
4952
4953
4954
4955
4956
4957
4958
4959
4960
4961
4962
4963
4964
4965
4966
4967
4968
4969
4970
4971

Implicit messages: As for soap-http-1.

Faults: As for soap-http-1.

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

Limitations on BTP use: None

Other: As for soap-http-1

Example using SOAP + Attachments binding

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

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

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

  <soap:Header>

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

      </soap:Header>

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

    </soap:Envelope>
```

4972
4973
4974
4975
4976
4977
4978
4979
4980
4981
4982
4983
4984
4985
4986
4987

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

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

--MIME_boundary--
```

4988 Conformance

4989
4990
4991
4992
4993
4994
4995
4996
4997
4998
4999

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

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

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

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

Cohesive Superior	Composer (as Superior only) Sub-Composer Coordinator (as Superior only) Sub-coordinator
Atomic Superior	Coordinator (as Superior only)) Sub-coordinator
Participant	Inferior Enroller

5000
5001
5002
5003
5004

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

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

5005
5006
5007
5008
5009

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

5010 sends ENROL with the same inferior address and with the “reply-address” absent (because
5011 the Inferior in all transactions are dealt with by the same addressable entity), must not assume
5012 that the same is true of received ENROLs)
5013
5014

5014
5015
5016
5017
5018
5019

Part 3. Appendices

The glossary is the subject of issue 4

A. Glossary

Message	A datum which is produced and then consumed.
Sender	The producer of a message.
Receiver	The consumer of a message.
Transmission	The passage of a message from a sender to a receiver.
Endpoint	A sender or receiver.
Address	An identifier for an endpoint.
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. <div style="border: 1px solid black; padding: 2px; margin: 5px 0;"><i>PRF - suffix ? I've used "additional information"</i></div>
Actor	An entity which executes procedures, a software agent.
Application	An actor which uses the Business Transaction Protocol.
Application Message	A message produced by an application and consumed by an application.

Application Endpoint	An endpoint of an application message.
Operation	A procedure which is started by a receiver when a message arrives at it.
Application Operation	An operation which is started when an application message arrives.
Contract	Any rule, agreement or promise which constrains an actor's behaviour and is known to any other actor, and upon which any other knowing actor may rely.
Appropriate	In accordance with a pertinent contract.
Inappropriate	In violation of a pertinent contract.
Service	An actor, which on receipt of an application messages, may start an appropriate application operation. For example, a process which advertises an interface allowing defined RPCs to be invoked by a remote client.
Client	An actor which sends application messages to services.
Effect	The changes induced by the incomplete or complete processing of a set of procedures by an actor, which are observable by another contemporary or future actor, and which are made in conformance with a contract known to any such observer. This contract must state the countereffect of the effect, and is known as the countereffect contract. An effect is Completed when the change-inducing processing of the set of procedures is finished. [Need an indirect or consequential damage exclusion clause]
	<i>PRF - Sentence about countereffect contract doesn't fit well</i>
Ineffectual	Describes a set of procedures which has no effect.
Countereffect	An appropriate effect intended to counteract a prior effect.

Countereffect Contract	<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_<u> </u>-identifier	An identifier assigned to an Inferior which is unique within the scope of an Address-as-Inferior.
Atomic Business Transaction	A set of participants (which may have only one member), all of which will receive instructions that will result in a homogeneous outcome.
<i>or</i>	(Transitively, a set of operations, whose effect is capable of countereffect.)
Atom	An atom is identified by an atom identifier.
Atom Identifier	A globally unique identifier assigned to an atom.
	<p><i>PRF – abs msgs define as unambiguous in scope of its address-as-superior, I think.</i></p>

Coordinator	An actor which decides the outcome of a single atom, and has a lifetime which is coincident with that of the atom. A coordinator can issue instructions to a participant to prepare, cancel and confirm. These instructions take the form of BTP messages. A coordinator is identified by its atom's atom identifier. A coordinator must also have a BTPA to which participants can send BTP messages.
Address-as-Superior	The address used to communicate with an actor playing the role of an Superior
Address-as-Composer	The address used to communicate with a Composer by an application actor that controls its resolution. The messages that might be sent to or received from this endpoint are undefined.
Address-as-Inferior	The address used to communicate with an actor playing the role of an Inferior.
Identity-as-Superior	The combination of superior_ <u> </u> -identifier and Address-as-Superior of a given Superior.
Identity-as-Inferior	The combination of inferior_ <u> </u> -identifier and Address-as-Inferior of a given Inferior.

5020